VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Vizualizace obsahu webové prezentace

Visualization of Web Presentation Content

2015

Bc. Martin Haščák

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Martin Haščák**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Vizualizace obsahu webové prezentace
Visualization of Web Presentation Content

Zásady pro vypracování:

Cílem práce je zmapovat současné možnosti vizualizace dat na webu a navrhnout odpovídající způsoby zpracování a zobrazení struktury webových stránek a jejich obsahových prvků.

1. Seznamte se a popište současné trendy v oblasti vizualizace a prezentace dat, a to s ohledem na dostupné technologie v prostředí webu (HTML5, SVG, atd.).
2. Navrhněte způsob analýzy a zpracování informací o struktuře a obsahu daných webových stránek tak, aby výsledek mohl být použitý pro efektivní vizualizaci struktury webové prezentace. Navrhněte vhodné způsoby vizualizace a interakce.
3. Navržený přístup implementujte v prostředí moderních webových technologií vč. možnosti využití existujících vizualizačních knihoven.
4. Vytvořte případové studie dokladující význam zpracování a vizualizace obsahu webových prezentací.
5. Celý přístup, včetně implementace vizualizačního nástroje a případových studií zhodnoťte, a to především z pohledu praktického využití.

Seznam doporučené odborné literatury:

[1] Jacques Bertin: Semiology of Graphics: Diagrams, Networks, Maps. ESRI Press. 2010. ISBN 978-1589482616
[2] Stephen Few: Information Dashboard Design: The Effective Visual Communication of Data. O'Reilly Media. 2006. ISBN 978-0596100162
[3] Kurt Cagle: HTML5 Graphics with SVG & CSS3. O'Reilly Media. 2013. ISBN 978-1449304478

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Michal Radecký, Ph.D.**

Datum zadání: 01.09.2014
Datum odevzdání: 07.05.2015

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Souhlasím se zveřejněním této diplomové práce dle požadavků čl. 26, odst. 9 *Studijního a zkušebního řádu pro studium v magisterských programech VŠB-TU Ostrava.*

V Ostravě 7. května 2015 ..... Hasičák Martin .....

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 7. května 2015 ..... Hasičák Martin .....

Na tomto místě bych chtěl poděkovat vedoucímu své diplomové práce,
panu Ing. Michalu Radeckému, Ph.D. za poskytnuté rady, trpělivost a připomínky k obsahu a formě zpracování práce.

Abstrakt

Tato práce se věnuje aktuálním trendům v oblasti vizualizace dat na webu. Na začátku seznamuje čtenáře s procesem získávání a vizualizace dat. Následně nastiňuje různé vizualizační techniky a prvky interakce, které lze v rámci webu využít. Získané znalosti jsou poté využity k tvorbě a implementaci konceptu vizualizačního nástroje, jehož primární účel je vizualizovat strukturu webové prezentace. Práce v závěru zhodnotí význam vizualizace dat v prostředí webu.

Klíčová slova: ASP.NET MVC, Canvas, D3.js, SVG, vizualizace, web

Abstract

This thesis explores the latest trends pertaining to data visualizations for the web. The beginning of the theoretical part offers coverage on the process of data mining and visualization. This is followed by the outline of various visualization techniques along with interaction elements which may be used within the web. The knowledge gathered from the theoretical background are then applied to the development and implementation of the visualization tool concept which primarily aims to visualize the structure of the web presentation. At the conclusion of the thesis, the significance of data visualization within the web environment is assessed.

Keywords: ASP.NET MVC, Canvas, D3.js, SVG, visualization, web

Seznam použitých zkratek a symbolů

AJAX	–	Asynchronous JavaScript and XML
API	–	Application Programming Interface
DOM	–	Document Object Model
JSON	–	JavaScript Object Notation
MVC	–	Model View Controller
ORM	–	Object Relational Mapping
REST	–	Representational State Transfer
SVG	–	Scalable Vector Graphics
URL	–	Uniform Resource Locator
WWW	–	Word Wide Web

Obsah

Seznam tabulek

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

S neustálým rozvojem informačních technologií a internetu se zvyšují nároky na přenos většího objemu digitálních dat. V průběhu dvou desetiletí se rychlost internetu a jeho dostupnost velmi rychle rozšiřuje po celém světě a tím narůstá i objem dat, která jsou produkována, přenášena a ukládána. Zdrojem dat dnes již nemusí být jen telefon či počítač, ale i chytré televize, hodinky, internetové stránky nebo různé snímače. Většina těchto zařízení jsou přímo nebo bezdrátově připojena k internetu, s jehož pomocí se strukturovaná data ukládají do datových skladů. Data jsou následně vyhodnocována a získané informace pak využity pro různé marketingové účely, vylepšení daného produktu apod. Aby analýzy byly přesnější, klade mnoho firem větší nároky na zpracování tzv. nestrukturovaných dat, která se v našem okolí vyskytují mnohem častěji. Jejich objem je však mnohonásobně větší, a proto je daleko náročnější i nákladnější tato data zpracovat a hledat v nich klíčové informace. V tomto kontextu se začali data označovat jako „BigData“ (velká data), která se svým charakterem nepodobají klasickým datovým skladům. Složitost a provázání dat kladou vyšší nároky na jejich smysluplnou vizuální reprezentaci. Staré techniky vizualizace přestávají stačit a objevují se nové inovativní prvky. Grafy přestávají být statické obrázky, stávají se z nich komplexní animované komponenty s prvky interakce. Jejich úkolem je co nejpřesněji zachytit provázanost dat s velkým důrazem na jednoduché pochopení komplexního problému z uživatelského hlediska.

Nové vizualizační techniky v rámci webu vyžadují nové nástroje a postupy pro datovou analýzu. S nástupem jazyka HTML 5 se díky elementu Canvas či SVG výrazně rozšířily možnosti vizualizace dat v prostředí prohlížeče. Při výběru vhodného nástroje je však nutné brát ohled na množství dat, která je nutno vizualizovat. Množství dat však stále přibývá, proto se reprezentace velkých dat stává klíčovou v mnoha oblastech a pojem BigData je stále častěji skloňován.

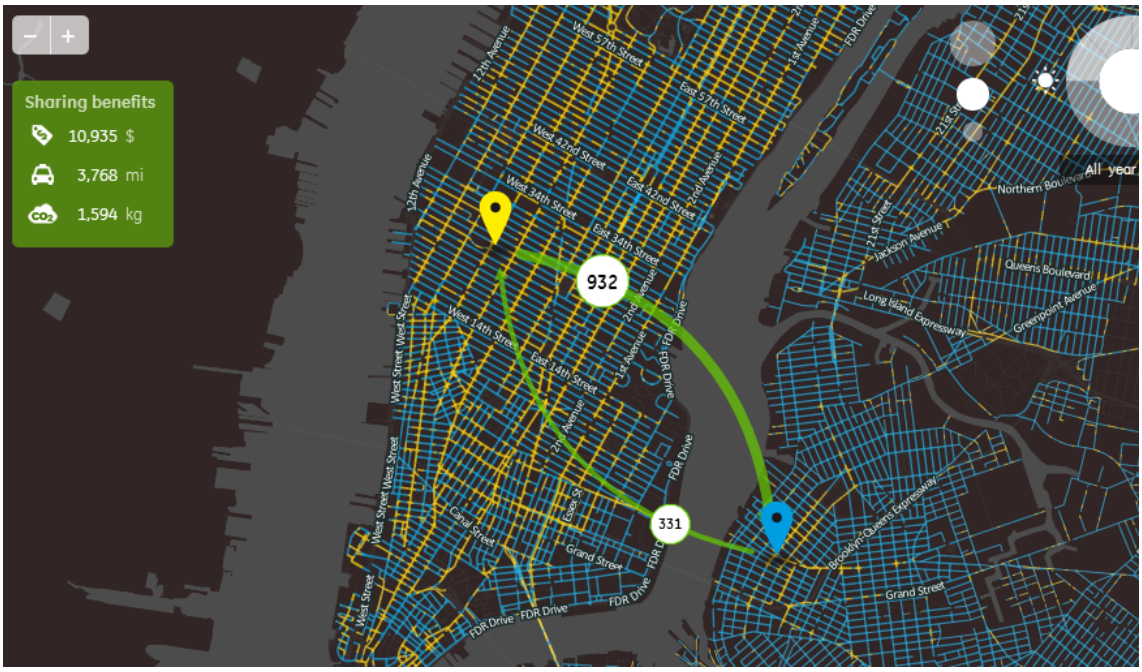
Definici pojmu BigData nejvýstižněji uvádí poradenská firma Gartner jako: „termín aplikovaný na soubory dat, jejichž velikost je mimo schopnosti zachycovat, spravovat a zpracovávat data běžně používanými softwarovými nástroji v rozumném čase“.[1]

Ačkoliv existuje více definic pojmu BigData, většina společností se shodla na základních charakteristikách, které jsou pro tento typ dat typické na rozdíl od klasického datového skladu. Např. firma IBM tato data charakterizuje pomocí trojce:

- **volume** – množství dat,
- **velocity** – rychlost jakou data přibývají,
- **variety** – různorodost dat.

Tento základní 3 dimenzionální model BigData je často doplňován o další charakteristiky, nejčastěji je však zmiňována *Veracity*, která se především zaměřuje na úplnost a důvěryhodnost dat [2].

Spojením vizualizačních technik a dostupných nástrojů v rámci webu lze vytvářet interaktivní aplikace, které jsou schopny zobrazit milióny záznamů. Příkladem takové aplikace může být projekt *HubCab*[1]. Zde tým datových, vizualizačních a webových profesionálů, sponzorovanými společnostmi jako je Audi, GE, vytvořil aplikaci, která vizualizuje cesty téměř 170 miliónů taxíků ve městě New York. Nejedná se ovšem o komerční aplikaci k běžnému použití, ale o pouhou ilustraci jak mohou BigData v souvislosti s pokročilými technologiemi sledovacích systémů přispět k úspoře paliva a emisí ve velkých městech. Vše funguje pomocí interaktivní mapy, kde si uživatel může označit místo odjezdu a cílovou destinaci, na kterou by se chtěl dostat taxíkem. Aplikace uživateli pak sama zobrazí, kolik lidí jezdí tu samou trasu v danou hodinu jako on. Zároveň ho na základě statistických dat informuje, kolik by se ušetřilo peněz a emisí, pokud by jízdu sdílel s lidmi, kteří mají stejnou cestu. Prostředí aplikace ilustruje obr. 1.



Obrázek 1: Projekt HubCab - ilustrace prostředí aplikace

Cílem této práce je vytvořit čtenáři náhled na současné metody vizualizace informací v kontextu webových stránek. Dále navrhnout způsob vizualizace a analýzy struktury webu s ohledem na jeho obsahové prvky. Výstupem této práce bude ukázková aplikace, která názorně demonstruje analýzu a vizualizaci obsahu různých webů ve smyslu firemních či osobních prezentací, internetových magazínů nebo obchodů apod.

[1]Projekt je dostupný na adrese http://hubcab.org

2 Data a jejich vizuální reprezentace

Data jsou vytvářena a zaznamenávána různými způsoby. Může se jednat o reprezentaci množiny čísel nebo slov, jejichž správnou interpretací je možné získat nějakou informaci. V oblasti výpočetní techniky jsou data zakódována do binární podoby (jako jedničky a nuly), které jsou uloženy pro případné další zpracování někde na paměťovém médiu. Vhodnou organizací, zpracováním a následnou prezentací dat v určitém kontextu lze získat informaci. Příkladem tzv. surových dat v kontextu webových stránek může být např. text článku. Jeho analýzou lze získat informaci o počtu slov, jenž článek obsahuje nebo o četnosti výskytu daného slova v článku. Aby získaným informacím uživatel jednoduše porozuměl a byl schopen informace analyzovat, je vhodné data vizualizovat pomocí tabulek, grafů či jejich kombinací.

Vizualizace dat se s růstem generovaných dat postupně stává poměrně komplexní disciplínou, která vyžaduje účast odborníků s různým zaměřením. Aby bylo možné data vizualizovat je nutné, aby jim člověk nejdříve porozuměl a věděl, co v nich hledá a proč. Může se totiž stát, že z jednoho datového souboru lze vydedukovat zcela odlišné výsledky, jelikož se můžeme dívat na stejné věci s jiným pohledem. Z tohoto hlediska je analýza dat náročnější na předchozí znalosti člověka, který s nimi pracuje.

Jak postupovat při vizualizaci dat od jejich získání, zpracování až po následnou interpretaci, zobrazuje diagram na obr. 2. Význam jednotlivých kroků je následně dále vysvětlen.[3]

Obrázek 2: Proces vizualizace dat [3]

- **Získávání dat** - základní fáze, nejdříve je nutné data vytvořit, stáhnout, najít z různě dostupných zdrojů (čidla, sociální sítě, GPS systémy apod.).
- **Zpracování** – transformace získaných dat do formátu vhodného pro strojové zpracování. Může obsahovat filtrování obsažené informace s ohledem na rychlost a snadnost následujícího zpracování.
- **Filtrování** – přímo navazuje na zpracování. V této fázi se třídí data na základě kritérií, kterým se chceme věnovat a na data, která pro nás nemají význam. Často se používají vícestupňové filtry pro získání dat, o která opravdu máme zájem.
- **Dolování** – tato fáze se soustředí na získávání informací z klíčového datového souboru. Většinou jsou zde využity nástroje, jako jsou např. regulární výrazy nebo matematická statistika. Jedná se o podstatnou část zpracování, kde se vytváří algoritmy, které dohledávají souvislosti v rámci souboru dat.

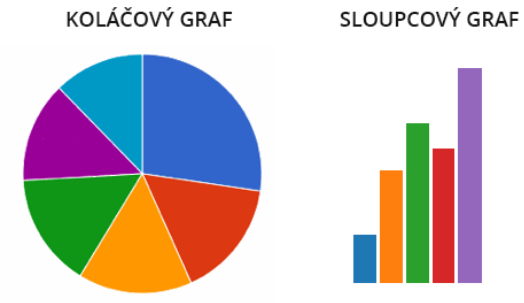
- **Reprezentace** – získaná data je třeba nějakým způsobem vizualizovat tak, aby byla snadno pochopitelná. Přesně to se děje v této fázi. Vybíráme vhodné vizualizační nástroje např. tabulky, grafy aj.
- **Čištění** – tato fáze ještě dále vylepšuje celkový vzhled dat, která chceme prezentovat. Například výběr vhodných barev pro označení zobrazených hodnot grafem. Znovu zde dochází k odstraňování nepotřebných dat.
- **Interpretace** – poslední fáze se soustřeďuje na hledaní cest pro nejlepší možné pochopení obsahu. Může jít například o přidání vrstvy zobrazující problém z jiného pohledu, animace v čase atp. Této fáze by se měl účastnit odborník, jenž se věnuje problematice, kterou data mají prezentovat.

2.1 Dimenze nad daty

Při vizualizaci informací se lze setkat s velkým množstvím datových záznamů různého typu. Každý záznam se může dále skládat z více atributů. Data spadající do stejné kategorie informací představují tzv. dimenzi. Například záznamy typu rok, měsíc, týden, den apod. spadají pod jednu časovou dimenzi. Počet dimenzí nad daty se v různých datových souborech může lišit. Data se proto podle dimenzí mohou dělit na jednodimenzionální, dvoudimenzionální až n-dimenzionální, která jsou rovněž označována jako multidimenzionální data[4].

2.1.1 Jednodimenzionální data

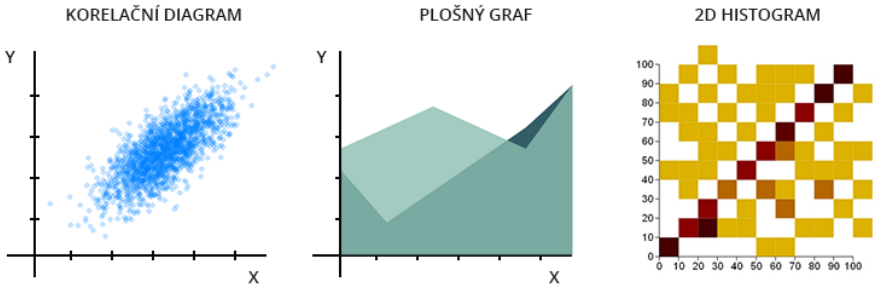
Datový soubor, jenž je tvořen jednolitými daty.[4] Typickým příkladem těchto dat jsou např. data vzniklá při měření konkrétní veličiny. Tato data lze rovněž jednoduše převést na dvoudimenzionální, např. prostým přidáním indexu u každého záznamu. Takový soubor dat lze znázornit třeba pomocí histogramu nebo koláčového grafu.



Obrázek 3: Příklady metod vizualizace jednodimenzionálních dat

2.1.2 Dvoudimenzionální data

Dvoudimenzionální data obsahují dvě odlišené dimenze. Typickým zástupcem tohoto druhu dat jsou geografická data, kde dvě odlišné dimenze tvoří souřadnice zeměpisné šířky a délky.[4] Každý bod, svázaný oběma souřadnicemi, je asociován s nějakou hodnotou v rámci datového souboru.[4] Dalším příkladem může být informace o rozlišení obrazovek návštěvníků webových stránek. Takové informace by se pak daly jednoduše vizualizovat pomocí bodového grafu.



Obrázek 4: Příklady metod vizualizace dvoudimenzionální dat

2.1.3 Multidimenzionální data

Mnoho datových souborů, které se skládají z více něž tří atributů reprezentující dimenzi, již není možné vizualizovat pomocí dvou nebo třídimenzionálních grafů. Relační databáze, čítající desítky či stovky sloupců - atributů, jsou klasickým příkladem multidimenzionálních dat. Vzhledem k tomu, že již není jednoduché mapovat atributy na dva rozměry obrazovky, je nutné k vizualizaci využít sofistikovanějších metod, např. metodu paralelních souřadnic.[4]

Data na webu, jako je text, multimediální obsah nebo hypertext jsou dalším příkladem multidimenzionálních dat, jež nelze jednoduše popsat pomocí čísel.[4] Pro jejich reprezentaci klasické vizualizační techniky již přestávají stačit. Vizualizací tohoto typu dat se proto dále věnuje kapitola 2.2.

2.2 Techniky vizualizace

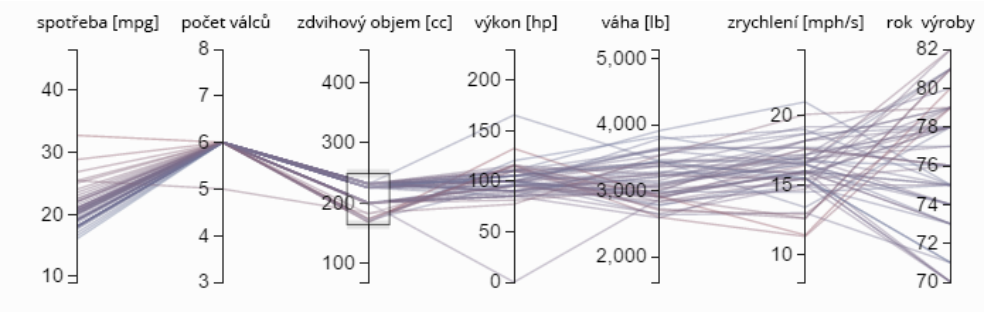
V případě potřeby vizualizovat již zpracovaný soubor dat nastane otázka: „Jakou vizualizační techniku použít tak, aby byl výsledek co nejvíce pochopitelný pro uživatele?" Většinou u této fáze zpracování dat musí být k dispozici odborník z vybrané oblasti zájmu. Tento člověk nebo skupina pak rozhodují o tom, jaká technika vizualizace je pro daný problém vhodná. Často však dochází k situaci, kdy pro analyzovaný problém žádná vi-

zualizace neexistuje a je nutné ji vymyslet buď kombinací existujících technik či vytvořením techniky nové.

Obecně lze podle Daniela Kima základní vizualizační techniky nad multidimenzionálními daty klasifikovat pomocí šesti tříd.[5]

2.2.1 Geometrická vizualizace

Geometrická vizualizace má za cíl nalézt zajímavé projekce vícerozměrných datových souborů. Tato třída vizualizace se proto zejména uplatňuje v oblastech statistické analýzy dat. Zahrnuje techniky jako je korelační diagram či paralelní souřadnice. Systém paralelních souřadnic analyzuje vícerozměrná data a mapuje je do 2 rozměrného prostoru, kde každou dimenzi reprezentuje jedna vertikální osa. Pomocí paralelních souřadnic lze v rámci webu vizualizovat například katalog produktů [2]. Každou osu by pak tvořil parametr, podle kterého by se produkty v katalogu mohly filtrovat, např. váha, výkon nebo rok výroby, jak ukazuje ilustrace na obr. 5.



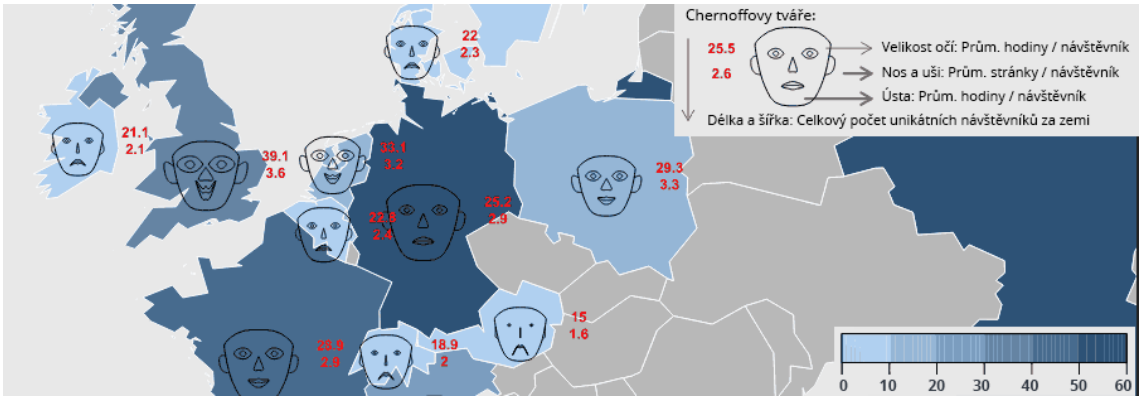
Obrázek 5: Vizualizace pomocí paralelních souřadnic [7]

2.2.2 Piktografická vizualizace

Cílem piktografické vizualizace je mapovat multidimenzionální datové položky na ikonky. Názorným příkladem může být technika *Chernoffovy tváře*, kterou navrhl pan Herman Chernoff v roce 1973.[8] Zde se využívá charakteristických rysů tváře jako je tvar nosu, uší a samotný tvar obličeje k vyjádření hodnot jednotlivých proměnných. Počet proměnných, které lze v rámci jedné tváře zaznamenat, je však omezený rozlišovacími schopnostmi člověka. Vizualizace se s jejich nárůstem stává obtížně čitelná. Piktografické vizualizace jsou limitovány rovněž prostorem a velikostí piktogramů, jenž se mohou začít s jejich nárůstem překrývat. Existují však piktografické techniky jako je *Stick Figures* nebo *Color Icons*, které se zobrazením vyššího množství objektů současně nemají problém [6].

[2]Studie od IBM, která se zabývá vizualizaci katalogu pomocí paralelních souřadnic je k dispozici zde: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.201.4437

Piktografické vizualizace lze uplatnit i v souvislosti s webovými daty. Konkrétně při analýze návštěvnosti webů lze využít techniku *Chernoffových tváří*, kde porovnáním tváří by šlo např. vyčíst, jakou dobu uživatel strávil na konkrétním webu, kolik navštívil stránek, počet unikátních návštěvníků apod.



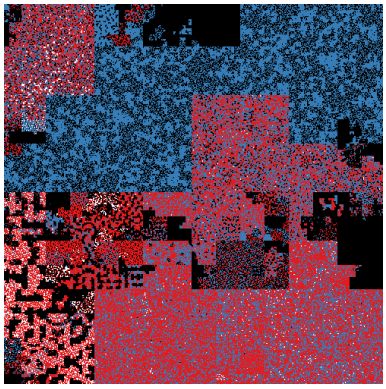
Obrázek 6: Vizualizace využití internetu v rámci Evropy pomocí Chernoffových tváří za rok 2012 [9]

2.2.3 Pixelově orientovaná vizualizace

Základní myšlenkou této třídy vizualizace je mapovat každou hodnotu datového atributu na barevný pixel. Hodnoty atributů pro každý atribut jsou pak reprezentovány samostatným pod-oknem.[10] Je zřejmé, že tato metoda je omezena množstvím pixelů na obrazovce, při rozlišení 1920x1080 lze tedy zaznamenat až 2 073 600 datových hodnot. Pixelově orientované techniky tak maximalizují množství informací, které lze zobrazit najednou bez vzájemného překrývání. Podle uspořádání pixelů lze tuto techniku rozdělit ještě na dvě kategorie.

- **Query-Independent** - technika vhodná pro přímou vizualizaci většího objemu dat. Třídí data na základě vybraných atributů a pro uspořádání pixelů na obrazovce využívá techniku prostor vyplňujících křivek (screen-filling).[10]
- **Query-Dependent** - zde se vizualizují data v kontextu uživatelsky specifického požadavku. Nejvíce relevantní data jsou pak uspořádána v centru okna, méně relevantní data jsou následně uspořádána do spirálovitého tvaru na vnějších stranách okna.[10]

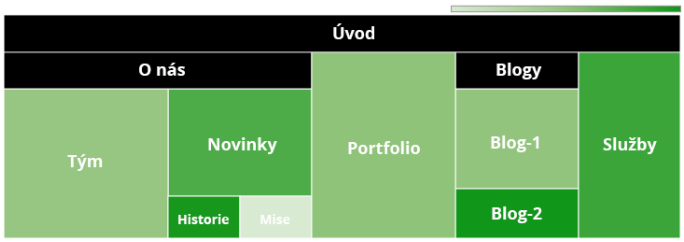
Na obr. 7 je ilustrace vizualizace binárního souboru s použitím pixelově orientované techniky a metodou vyplnění prostoru pomocí Hilbertových křivek. Modrá barva zde představuje tisknutelné znaky, černá barva hodnotu *0x00* čili neplatný znak, bílá barva hodnotu *0xFF* a červená všechno ostatní.



Obrázek 7: Vizualizace binárního souboru pomocí Hilbertových křivek [11]

2.2.4 Hierarchická vizualizace

Hierarchická data jsou data, která je možné reprezentovat pomocí stromu. Tato třída vizualizace se snaží rozdělovat multidimenzionální prostor na podprostory. [5] Existují dva základní přístupy pro vizualizaci hierarchií. První využívá strukturu uzlů a hran, které vyjadřují vztah mezi uzly. Druhý přístup je založen na efektivním vyplněný prostoru s ohledem na relativní velikosti uzlů v rámci hierarchie, jako to dělá technika tzv. uspořádaných map - *TreeMap* viz. obr. 8. Tato technika využívá strukturu do sebe zanořených obdélníků, které mají určitou velikost, umístění a barvu. Každou větev stromu reprezentuje jeden obdélník, jenž je postupně doplňován o další obdélníky, pokud se větev dělí na další pod-větve.[12] Použití této metody je výhodnější v případě nutnosti zobrazit hierarchickou strukturu s ohledem na úsporu prostoru vyhrazeného pro vizualizaci.

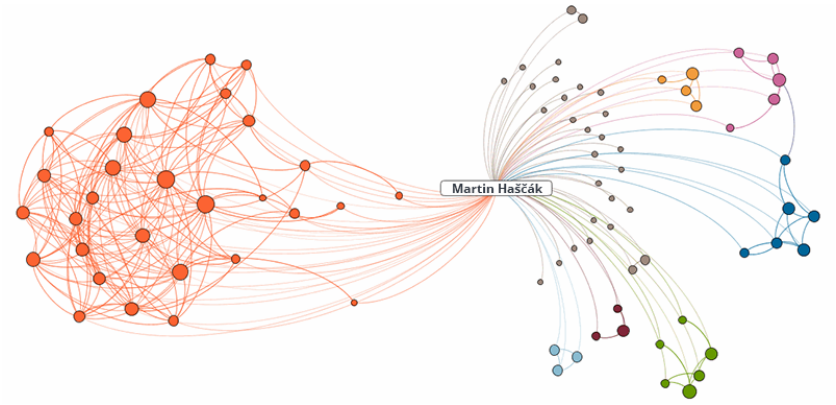


Obrázek 8: Vizualizace struktury webu pomocí techniky uspořádaných map

2.2.5 Grafová vizualizace

Třída zaměřená na vizualizaci množiny objektů, které mohou být vzájemně mezi sebou propojeny. Množinu objektů pak dle teorie grafů reprezentují uzly nebo vrcholy grafu a vazby mezi objekty jsou znázorněny jako hrany propojující každé dva uzly. Třída grafových vizualizací zahrnuje množství technik, které se zaměřují na efektivní rozložení uzlů

a hran tak, aby smysl vizualizace datového souboru byl pochopen rychle a jasně. Často se tento způsob vizualizace používá při analýze vazeb v rámci sociálních dat, jako to ukazuje obr. 9. Barevně odlišené uzly a hrany v obrázku představují různé vztahy nebo skupiny v rámci profesní sítě daného uživatele.



Obrázek 9: Vizualizace struktury profesní sítě LinkedIn vybrané osoby

2.2.6 Hybridní vizualizace

Do této třídy patří metody kombinující více vizualizačních technik najednou nebo zobrazující více pohledů na data v různých oknech za účelem lepšího pochopení celku [5]. Vzájemným spojením vizualizací lze uživateli poskytnout více informací, proto se hybridní vizualizace běžně používají i v rámci webu většinou s vazbou na dynamickou interaktivní stránku.

3 Vizualizační nástroje

Na internetu je dostupných mnoho vizualizačních nástrojů[3] a knihoven[4], které implementují zmíněné techniky vizualizace s vazbou na dostupné metody tvorby vizualizací v rámci webu. Každý z nástrojů je buď specificky zaměřen na konkrétní typ dat nebo poskytuje možnost vizualizovat data z různých problémových oblastí, např. sociální sítě, bioinformatika, sémantický web.

Podle fyzického umístění lze rozdělit vizualizační nástroje do tří skupin:

- Knihovny v podobě hotového zdrojového kódu, které lze využít na vlastních webových stránkách, viz první případ na obr. 10.
- Aplikace na straně klienta generující samostatné vizualizace v různých formátech, např. SVG, PDF, viz druhý případ na obr. 10.
- Speciální cloudové aplikace, které vytvářejí vizualizace na základě nahraných dat jako službu, viz třetí případ na obr. 10.

Obrázek 10: Rozdělení vizualizačních nástrojů

3.1 Vizualizace v prostředí webu

Cílem vizualizace je jednoduše porozumět datům za pomocí lidského vizuálního systému. Dobře navržená vizuální reprezentace dat může nahradit kognitivní výpočty jednoduchým vnímáním a vyvodit tak závěry poměrně složitého problému za pár sekund. Vizualizace se stává vizualizací, až ji někdo spatří na vlastní oči. Podle statistik mělo již v roce 2013 počítač s připojením k internetu průměrně 75% obyvatel EU.[13] Publikace vizualizací v prostředí webu proto dává smysl a jedná se snad o nejrychlejší způsob, jak

[3]Některé z nich jsou zmíněny např. zde http://www.hongkiat.com/blog/data-visualization-tools-resources/

[4]Některé z nich jsou zmíněny např. zde http://www.fastcolabs.com/3029760/the-five-best-libraries-for-building-data-vizualizations

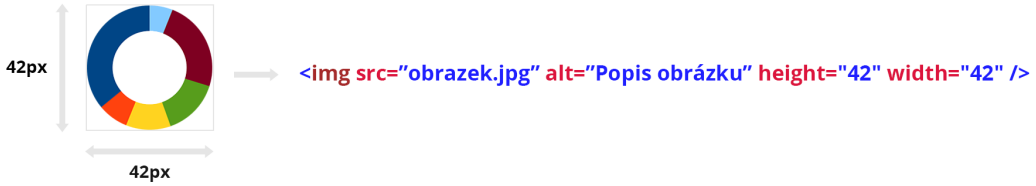
dosáhnout globálního publika. V dnešní době existuje stále větší a větší počet nástrojů s integrovaným webovým prohlížečem, jako jsou např. tablety, mobily nebo televize. Začíná být trendem mít připojení k internetu dostupné kdekoliv, kdykoliv a v čemkoliv, např. lednice, automobil, inteligentní dům. Počet těchto inteligentních zařízení s rychlým rozvojem vědy/techniky stále narůstá a v kombinaci s webovými standardy vytváří ideální platformu pro vizuální reprezentaci dat.

V prostředí webu lze vytvářet jednak statické vizualizace či vizualizace interaktivní. Statické vizualizace představují jeden konkrétní pohled na data. Více těchto statických pohledů pak může reprezentovat stejná data z různé perspektivy. Pokud je potřeba vizualizovat data v konkrétní čas na jednom místě, je počet dimenzí nad daty rovněž omezen. Reprezentace multidimenzionálních dat pomocí statických obrázků je tak velmi obtížná.

Dynamická vizualizace dat nabízí možnost zkoumat data do větší hloubky a z různých pohledů současně. Mezi základní funkce většiny interaktivních vizualizací patří přiblížení, oddálení nebo dynamické načítání detailních informací na vyžádání. Tyto základní funkce jsou doplňovány dalšími prvky, jako je animace, gravitace či detekce kolizí a nabízí tak uživateli komfortní cestu k pochopení datového celku.

V rámci webových standardů existuje pět obecných způsobů jak data vizualizovat. Každý z nich je více či méně vhodný pro různé techniky vizualizace. Stručný přehled způsobů (jejich výhody a nevýhody) je popsán v následujících bodech.

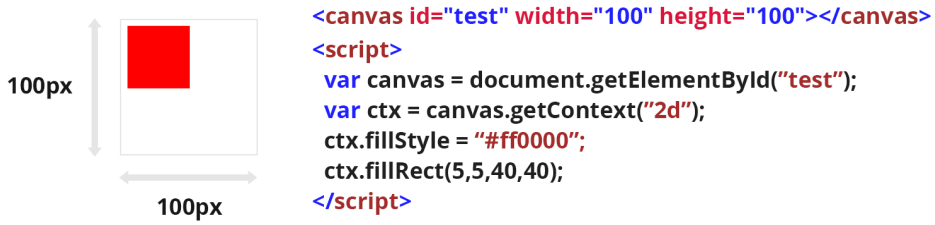
- **Obrázek** - jeden ze základních, nýbrž dodnes často používaných způsobů vizualizace. Pomocí HTML elementu <*img*> lze zobrazit jakýkoliv statický graf jako obrázek, který může být vygenerován dynamicky na základě dat uložených na serveru. Jeho nevýhodou je úplná absence interakce a závislost na serveru. Může se hodit k vykreslení jak základních, tak pokročilejších grafů, jako je sloupcový, boxový, koláčový graf nebo mapa. Absence interakce umožňuje data zobrazit jen z jednoho pohledu.



Obrázek 11: Příklad vizualizace pomocí elementu img

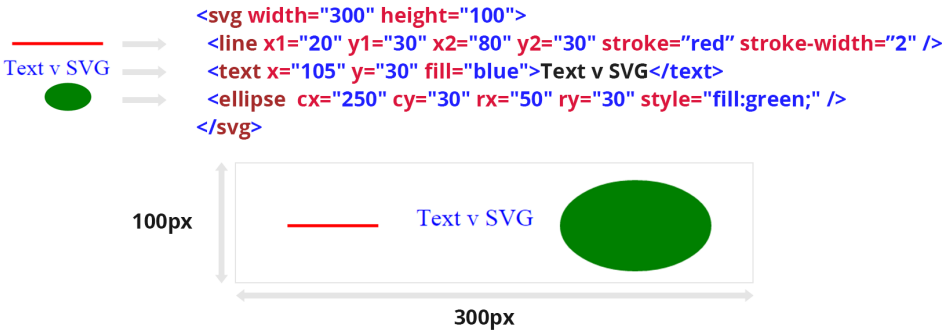
- **HTML element CANVAS** - nový element, který byl představen s příchodem HTML5. Doplňuje ho JavaScriptové API, s jehož pomocí kontejner umožňuje programově vykreslovat různé druhy grafiky od jednoduchých linií až po komplexní grafické objekty. Základní vlastností tohoto elementu je nutnost vykreslovat daný objekt pixel po pixelu. Pokud se pozice objektu změní, je nutné překreslit celé plátno. Díky

dostupnému API je možné do vizualizace přidat prvky interakce a hodí se tak pro dynamické vykreslování dat z více pohledů. Výhodou může být nezávislost na internetovém připojení a podpora hardwarové akcelerace.



Obrázek 12: Příklad vizualizace pomocí Canvas

- **SVG** - značkovací jazyk založený na syntaxi XML. SVG, umožňuje kreslit vektorovou grafiku přímo v prohlížeči na rozdíl od HTML 5 Canvas, který pro vykreslování využívá techniku rastrování.[16] Každý vizuální objekt je v SVG popsán pomocí XML a uložen v rámci DOM webové stránky.[16] Jelikož je každý objekt součástí stránky, je možné s ním jednoduše manipulovat a nevázat na něj pomocí JavaScriptu různé události. Díky skriptování a možnosti manipulace s DOM lze pomocí SVG vytvářet interaktivní vizualizace nezávislé na rozlišení obrazovky. SVG se zejména využívá např. pro tvorbu mapových podkladů.



Obrázek 13: Příklad vizualizace pomocí jazyka SVG

- **HTML DOM elementy** - jednoduché vizualizace, jako je sloupcový či koláčový graf, mohou být vytvořeny pouze za pomocí HTML a kaskádových stylů. S využitím CSS lze kontrolovat pozice na x/y souřadnicích a zároveň na základě dostupných údajů procentuálně nastavovat výšku/šířku např. řádku nebo sloupce. Díky CSS lze do grafu zanést i jistou míru interakce a s využitím CSS 3 i jednoduché animace.

- **WebGL** - pravděpodobně nejefektivnější nástroj pro vykreslování grafiky v rámci webu. WebGL je technologie, která umožňuje s pomocí hardwarové akcelerace vy-

HTML

```
<ul class="chart">
  <li>
    <span style="height:35%" title="SQL"></span>
  </li>
  <li>
    <span style="height:10%" title="PHP"></span>
  </li>
</ul>
```

Výstup

CSS

```
.chart {
  display: table;
  width: 60%;
  height: 200px;
  margin: 0 auto;
}

span {
    margin: 0 1em;
    display: block;
    background: red;
}

.chart  li {
    position: relative;
    display: table-cell;
    vertical-align: bottom;
    height: 200px;
  }

span:before  {
      position: absolute;
      left: 0; right: 0; top: 100%;
      display: block;
      text-align: center;
      content: attr(title);
    }
```

Obrázek 14: Příklad vizualizace pomocí HTML / CSS

kreslovat v prohlížeči 3D grafiku bez nutnosti instalace dalších doplňků. Ke své činnosti využívá HTML element Canvas. WebGL program je rozdělen na řídící kód napsaný v JavaSriptu a shader kód napsaný v jazyku GLSL, jenž je vykonáván grafickou kartou počítače. Shader kód může být definován přímo v HTML pomocí tagu <*script*> nebo v JavaScriptu jako řetězec.[17][18]

4 Analýza struktury webu a možnosti vizualizace

Webové stránky se skládají z různých typů informací a dat, které jsou vytvářeny jednak jejich tvůrci, ale především uživateli, jenž stránku navštěvují. Zaznamenání vzájemné interakce mezi návštěvníky a webovou prezentací dále doplňuje široké spektrum informací, které lze v dnešní době z webu vyčíst. Tato data, pokud je dokážeme správně analyzovat a vizualizovat, představují zlatý důl pro vlastníka webových stránek. Na jejich základě se totiž může web dále úspěšně rozvíjet.

Aby bylo možné data vizualizovat, je nutné nejdříve najít způsob, jak je z webu efektivně vyčíst. Tato kapitola popisuje způsoby dolovaní dat z webu s cílem navrhnout vhodné způsoby vizualizace a interakce s těmito daty.

4.1 Jaká data jsou zajímavá

Každá stránka je svých obsahem a strukturou specifická. Obsahuje obrázky, odkazy, různé reklamní bloky, texty aj. Tyto informace jsou zapsány v kódu HTML pomocí speciálních tagů a interpretovány webovým prohlížečem do ucelené vizuální podoby s využitím kaskádových stylů. Aby se web v prohlížečích správně vykresloval, musí jeho tvůrce dodržovat sadu pravidel pro specifický jazyk, který k jeho tvorbě používá (nejčastěji HTML, CSS, JS). Jako první by se logicky mohlo analyzovat, zda webová prezentace dodržuje standardy technologií, na nichž je postavena. Ovšem má smysl se zabývat analýzou těchto dat? Aktuálně se weby často skládají z různých šablon, kde např. hlavička a patička jsou pro všechny stránky stejné. Jedna chyba v kódu by se projevila na všech stránkách webu. Ve výsledku by tedy žádná stránka nebyla validní.

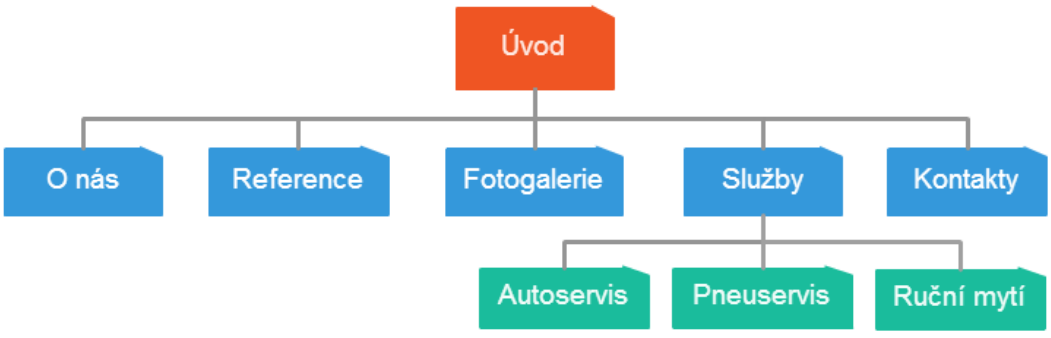
Aby bylo možné web efektivně popsat, je třeba znát jeho strukturu. Existuje několik základních typů struktur, které se při vývoji webových stránek více nebo méně používají [14]:

- **Lineární struktura** - základní struktura, která uvažuje sekvenční navigaci podobně jako při ovládání PowerPointové prezentace. Základní myšlenkou je fakt, že se nelze dostat na další stránku bez navštívení stránky předchozí. Tato struktura je vhodná pro popis postupu krok za krokem, např. objednávkový proces, nebo se může uplatnit u různých dotazníkových aplikací.
- **Paprsková struktura** - zde se předpokládá existence centrálního uzlu - stránky, která obsahuje odkazy na své podstránky. Z podstránek se ovšem dá dostat pouze zpět na hlavní stranu. Vzájemným nepropojením podstránek lze zaručit návrat na hlavní stránku, např. až při splnění daného úkolu.
- **Síťová struktura** - tato informační struktura webu zajišťuje to, že z každé stránky se lze dostat na všechny ostatní stránky a naopak. Tento případ často nastává, když u menšího webu je k dispozici hlavní navigační panel, např. v hlavičce nebo patičce stránky.

- **Hierarchická (stromová) struktura** - struktura typická pro většinu webových stránek. Jednotlivé stránky jsou uspořádány ve vztahu rodič a potomek. Příkladem mohou být různé katalogy nebo kategorie v internetovém obchodu.
- **Polyhierarchická struktura** - platí zde totéž co u předchozího případu, avšak potomek může mít více rodičů. S tímto rozložením se lze dnes setkat u většiny webů.

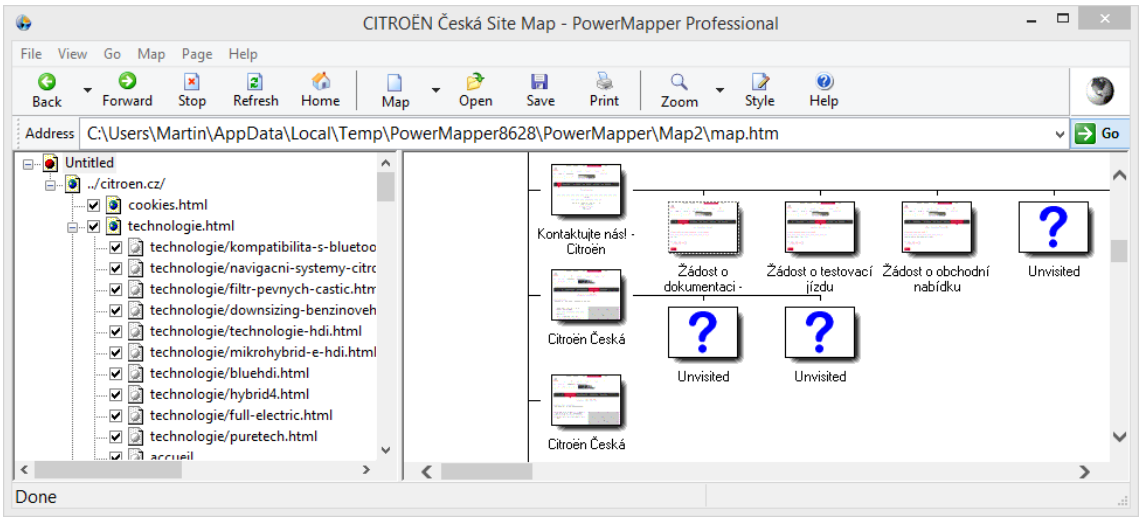
Vhodná grafická reprezentace této struktury u konkrétního webu může pak tvořit základ pro jednoduchou orientaci uživatele pomocí navigace, ale i jako výchozí bod pro jeho hlubší analýzu. Aby bylo možné strukturu webu vizualizovat, je nutné nejprve údaje o jeho struktuře získat a zpracovat do podoby vhodné pro jejich vizualizaci. Ještě před tím je ale dobré zmínit některé existující způsoby a nástroje pro vizualizaci struktury webu a posoudit, zda jsou pro představu detailnější analýzy použitelné.

Jednou ze základních vizualizačních technik struktury webu je tzv. mapa stránek. Její účel je především ulehčit orientaci návštěvníkům webu stromovým výpisem všech stránek, na které se v rámci prezentace může uživatel dostat. Mapa stránek může mít více podob, nejčastěji její vzhled ilustruje více úrovňový seznam na obr. 15. Mapa stránek v jiném formátu může být užitečná i pro vyhledávací roboty, kteří mají k dispozici na jednom místě odkazy na všechny stránky webu. Tímto se podstatně ulehčí jejich práce při indexaci webového obsahu. Často jsou tyto mapy tvořeny ručně nebo pomocí různých online generátorů, které používají jednoduché pavouky (crawlery). Ti rekurzivně prochází a ukládají URL odkazy v rámci konkrétní domény webu a rovnou generují výstup v požadovaném formátu. Generátory často neberou v potaz další dostupné informace na webu a své výstupy si neukládají pro další použití. Jedná se tady o omezené nástroje nevhodné pro komplexnější analýzu struktury webu. Pokud by generátory měly paměť, umožnilo by to analyzovat vývoj stránek v čase a např. s vazbou na další analytické nástroje by mohly svůj výstup obohatit o cenné informace jako je návštěvnost, průměrná doba zobrazení stránky, míra opuštění a další. Výstupem by pak mohla být jak vizuální mapa stránek pro uživatele, tak mapa stránek pro roboty v jiném formátu doplněna o další atributy upřesňující důležitost stránky.



Obrázek 15: Mapa stránek

Existují však i sofistikovanější nástroje, jenž zmíněné nedostatky generátorů řeší a doplňují je o další funkce. Příkladem může být nástroj *PowerMapper* nebo Site Vizualizer od společnosti *Elphsoft*. PowerMapper dokáže analyzovat a vizualizovat strukturu webu, přitom je schopný tato data propojit s dalšími analytickými nástroji, jako je např. Google Analytics. Kromě těchto nástrojů jich existuje ještě celá řada. Většinou jsou však úzce zaměřeny na specifická data a jsou zpoplatněny. Pro účely této práce bude tedy navržen a vytvořen nástroj vlastní.



Obrázek 16: Prostředí nástroje PowerMapper

4.2 Získání a zpracování dat

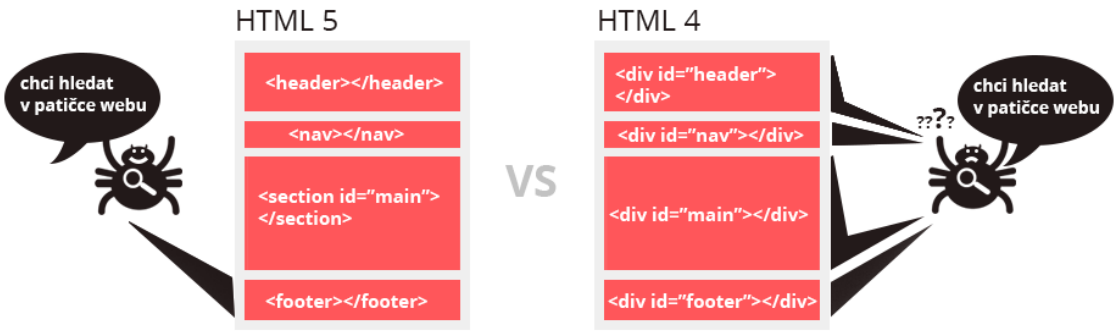
Jak již bylo zmíněno, struktura webu může být různá. Aby bylo možné tuto strukturu efektivně odhalit, je zapotřebí tzv. *crawler*. Jedná se o automatizovaného robota, jehož základním úkolem je na základě vstupní URL adresy identifikovat odkazy v rámci dané stránky. Robot následně rekurzivně prochází nalezené interní odkazy a zanořuje se tak hlouběji do struktury. Crawler by si měl pamatovat, jaké stránky již navštívil a pokračovat v prohledávání dokud nenavštíví všechny odkazy nebo ho nezastaví jiná podmínka. Při průchodu webem si crawler může zaznamenávat i informace vztahující se k jednotlivým stránkám, např. titulek, meta-tagy aj. a hlavně data o struktuře (odkud se kam dostal).

Dle standardu HTML lze k propojení webu využít následující elementy *a*, *area* nebo *link*. Element *link* se vyskytuje v hlavičce dokumentu a slouží převážně pro propojení stránky s externím zdrojem, jako je např. soubor s kaskádovými styly. Další element *a* s atributem *href* definuje tzv. hypertextový odkaz. Atribut *href* určuje cíl odkazu a umožňuje tak odkazovat z jedné stránky na druhou. Hodnotou atributu však nemusí být jen odkaz na jinou stránku, ale taktéž odkaz na soubor, emailovou adresu nebo na záložku.

Poslední element *area* se používá v kontextu elementu *map*. Tvoří aktivní oblast v rámci obrázkové mapy, a tak umožňuje v určitých místech obrázku klikat na odkazy, které mohou být opět různého typu.

Pro potřeby crawlera v kontextu webových stránek plně postačí analýza odkazů definovaných elementem *a* s atributem *href*, který se nejčastěji používá k navigaci v rámci webové stránky. Jelikož atribut může obsahovat různé typy odkazů, je potřeba, aby crawler při zpracování zohlednil pouze odkazy na stránky a soubory. To zahrnuje i odkaz na záložku, tzv. kotvu, která se využívá pouze v kontextu jedné stránky a odkazuje na určité místo v dokumentu.

Při procházení webu je vhodné zohlednit pozici nalezeného odkazu. S nástupem HTML 5 se začaly využívat nové sémantické elementy, které umožňují lépe popsat strukturu obsahu na stránce. Ačkoliv pro normálního uživatele nemají tyto sémantické elementy prakticky žádný význam, je vhodné je při návrhu stránky využívat. Ocení to zejména různí pavouci/roboti, kteří webovou stránku navštěvují a prohledávají za různým účelem. Většina internetových stránek je rozdělena do tří hlavních bloků, na hlavičku, obsah a patičku. Hlavička bývá umístěna v horní části stránky a obsahuje nejčastěji logo společnosti spolu s hlavní navigací. Blok obsahu umístěný ve středu stránky tvoří její hlavní část, která obsahuje primární informace či specifickou funkčnost pro návštěvníky. Poslední blok zpravidla obsahuje informace o autorovy, kontaktních údajích nebo informace o autorských právech.[15] Na základě tohoto pozorování vznikly zmíněné sémantické elementy <*header*>, <*nav*>, <*footer*> aj. Crawler díky sémantice může podle umístění odkazu poznat jeho význam a následně přizpůsobit své chování podle potřeby viz obr. 17.

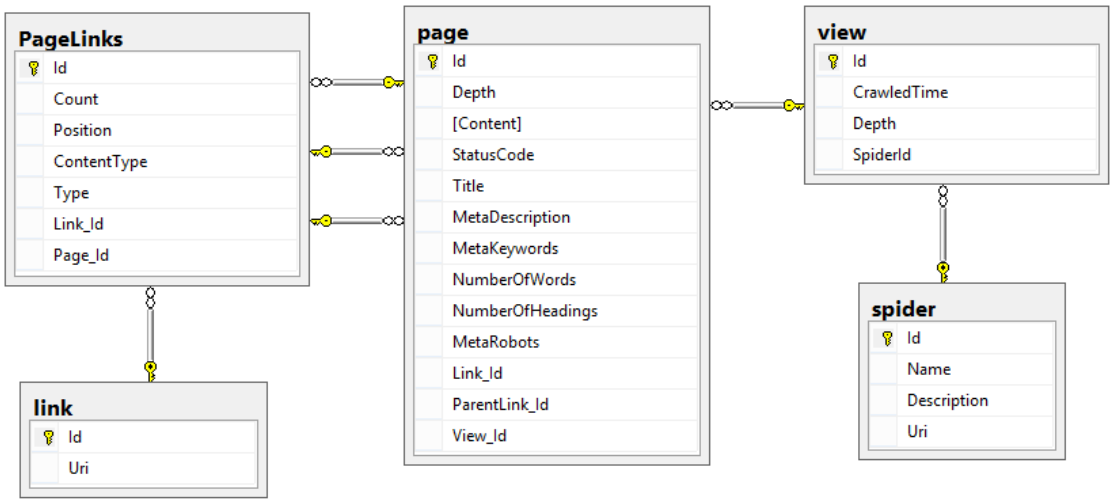


Obrázek 17: Sémantika HTML 5 vs HTML 4

Kromě zmíněné pozice odkazu je nutné uvažovat o faktu, že se daný odkaz může vyskytovat v rámci stránky na různých nebo stejných pozicích několikrát. Při prohledávání by měl crawler toto zohlednit např. tím, že pro každou stránku bude uvažovat jednoduché počítadlo. Crawler by tak při návštěvě stejného odkazu dále stránku nezkoumal, ale pouze inkrementoval počítadlo.

Aby bylo možné získaná data vizualizovat, je nutné je ve vhodné podobě zazna-

menávat v databázi. Struktura databáze je zobrazena na obr. 18. Jelikož web s každou návštěvou crawlera může mít jinou strukturu, je vhodné umožnit zaznamenat historii prohledávání. K tomuto účelu slouží tabulka *spider* a tabulka *view*. Tabulka *spider* identifikuje crawlerův vstupní bod, respektive URL adresu a tabulka *view* představuje konkrétní pohled na web v daném čase a z určité hloubky, do které se prohledáváním odkazů crawler zanořil. Tabulka pak *page* udržuje informace o konkrétní stránce a spolu se zbylými tabulkami rovněž informace o odkazech vyskytujících se v rámci stránky. Výsledný datový model je klíčovým prvkem aplikace a může být dále rozšiřován. Umožňuje vizualizovat strukturu webu v určitém čase, ale rovněž dává možnost doplnit vizualizaci o informace týkající se každé stránky.



Obrázek 18: Struktura databáze

Získané informace o struktuře by šlo ovšem rozšířit o další. V současné době existuje mnoho analytických nástrojů, které analyzují provoz webu někdy i v reálném čase. Příkladem takového nástroje je služba *Google Analytics*[5]. Ta po umístění JavaScriptového skriptu do HTML kódu stránky začne sbírat informace o webu a zasílat je ke zpracování na server. Zpracovaná data lze pak jednoduše vyexportovat v různých formátech nebo získat pomocí dostupného API řešení. Jelikož se zde data většinou vážou ke konkrétní URL adrese stránky, je možné je jednoduše propojit s daty získanými crawlerem při analýze struktury webu a ještě více tak obohatit výslednou vizualizaci.

4.2.1 Význam sloupců tabulek

Struktura databáze není nijak rozsáhlá, ovšem pro úplné pochopení navrženého schématu je dále popsán význam jednotlivých sloupců tabulek v následujícím pořadí: název sloupce, datový typ a krátký popis.

[5]Služba Google Anylytics je dostupná na adrese http://www.google.com/analytics/

Tabulka Page

`Depth`	int	Aktuální hloubka zanoření crawlera
`StatusCode`	int	Stavový kód HTTP serveru, který byl vrácen při dotazu
`Title`	VARCHAR	Titulek stránky
`MetaDescription`	VARCHAR	Popis obsahu stránky zadaný META atributem description
`MataKeywords`	VARCHAR	Klíčová slova na stránce zadaná META atributem keywords
`MetaRobots`	VARCHAR	Obsah meta atributu ROBOTS - ten říká, zda mohou roboti danou stránku zaindexovat
`NumberOfWords`	int	Počet slov na stránce, po odstranění HTML značek
`NumberOfHeadings`	int	Počet nadpisů na stránce
`View_Id`	int	Infromace k jakému pohledu stránka patří - cizí klíč
`ParentLink_Id`	int	Informace o rodiči - cizí klíč
`Link_Id`	int	URL adresa aktuální stránky - cizí klíč

Tabulka PageLink

`Count`	int	Počet odkazů na stejné pozici
`Position`	int	Pozice odkazu v rámci HTML: patička / navigace / obsah
`Type`	int	Typ odkazu interní / externí
`Link_Id`	int	URL odkazu - cizí klíč
`Page_Id`	int	Stránka, ke které odkaz náleží

Tabulka View

`CrawledTime`	DATETIME	Čas vytvoření nového pohledu na web
`Depth`	int	Nastavení zanoření crawlera - představuje ukončovací podmínku při prohledávání webu
`Spider_Id`	int	Informace, do jakého projektu daný pohled patří - cizí klíč

Tabulka Spider

`Name`	VARCHAR	Název projektu
`Description`	VARCHAR	Popis porojektu
`Uri`	VARCHAR	Vstupní URL adresa, od které začne crawler prohledávat web

Tabulka Link

`Uri`	VARCHAR	Absolutní adresa odkazu / stránky

4.3 Vizualizace a interakce

Z informací, získaných pomocí crawlera, lze modelovat strukturu webu jako orientovaný graf, který, jak již bylo zmíněno, může mít více podob. Každý uzel grafu by pak odpovídal navštívené stránce a orientovaná hrana mezi dvěma uzly by vyjadřovala vazbu mezi stránkami. Vzhledem k tomu, že crawler může nasbírat i doplňkové informace o každé navštívené stránce nebo o počtu odkazů na konkrétní pozici směřujících na jinou stránku, je nutné se zamyslet nad jejich vizualizací. Tyto informace mohou mít velký vliv na celkové chápání struktury dané webové prezentace. Kromě vizualizace je nutné zvážit i její dynamickou interaktivní stránku. Ta může mít naopak vliv na uživatelský komfort nebo na vyjadřovací schopnost vizualizace.

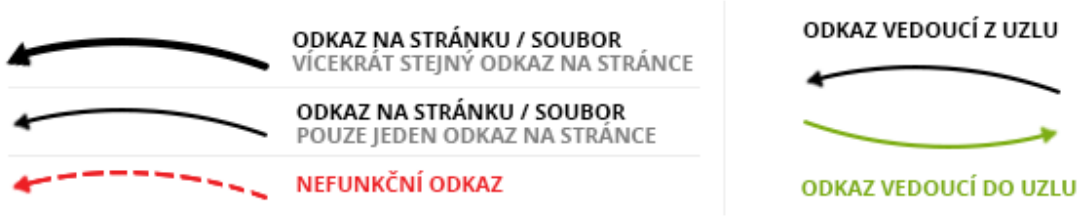
4.3.1 Vizualizace uzlu

Každý crawlerem navštívený odkaz představuje uzel grafu, který může být zobrazen ve formě kruhu. Poloměr kruhu, pokud se jedná o stránku, může určovat počet slov na stránce. Situace, kdy je odkaz nefunkční, či dojde k nějaké chybě, lze zohlednit barvou uzlu. Další informace vztahující se k jednotlivým uzlům mohou být získány až na vyžádání, např. kliknutím na uzel. Velikost obsahu u stránky může být různá, stránka s větším obsahem může čítat i tisíce slov. To bylo by tedy vhodné omezit maximální velikost uzlu tak, aby v některých případech nezabíral celou plochu vizualizace. Uzel může kromě HTML stránky vystupovat jako soubor či obrázek. Rozdíl mezi jednotlivými typy uzlů by šlo zohlednit pomocí obrázků. Možnosti vizualizace uzlu ilustruje obr. 19.

Obrázek 19: Vizualizace uzlu

4.3.2 Vizualizace hrany

Odkazy na stránce, směřující na jiné stránky nebo soubory, představují v rámci grafu orientované hrany. Vícenásobný výskyt stejného odkazu na stránce lze vizualizovat změnou tloušťky hrany. Podobně jako u uzlu je nutné tuto tloušťku limitovat. Nefunkční odkaz jde opět znázornit pomocí změny barvy hrany, alternativně přerušovanou křivkou. Informace o pozici odkazů do vizualizace ale zahrnout nejde, jelikož orientovaná hrana může představovat výskyt více odkazů na různých pozicích najednou. Výčet všech odkazů na stránce včetně informací o pozicích odkazů by bylo tedy vhodné zobrazit až na vyžádání u konkretního uzlu. Směr orientované hrany může být taktéž barevně odlišen, např. při najetí myší nad uzel. Možnosti vizualizace hrany ilustruje obr. 20.

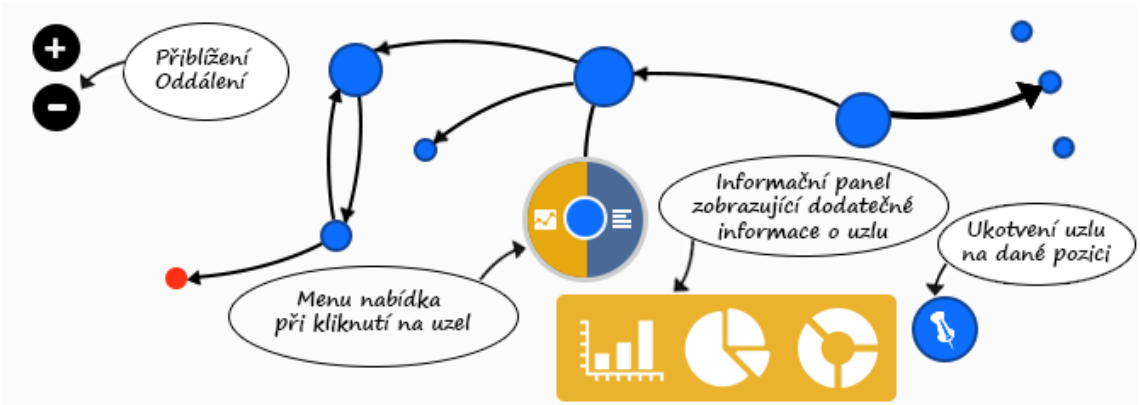


Obrázek 20: Vizualizace hrany

4.3.3 Návrh celkového konceptu

Díky možnosti spárovat URL adresy s daty dostupných z analytických nástrojů, alternativně daty, které může analyzovat crawler při průchodu webem, se naskytuje možnost vizualizaci struktury webu doplnit o další vizualizace. Příkladem informací, které se mohou vázat k dané stránce, je třeba statistika návštěvnosti, doba strávená na stránce, počet slov na stránce, počet nadpisů nebo míra okamžitého opuštění. Tyto informace lze vizualizovat pomocí klasických dvoudimenzionálních grafů, jako je např. koláčový či sloupcový graf. Vhodným zakomponováním těchto dílčích vizualizací včetně vizualizace struktury webu do jednoho celku lze uživateli nabídnout ještě vyšší přidanou hodnotu. Celkový navržený koncept vizualizace, včetně možných interaktivních prvků, je zobrazen na obr. 21.

Kromě vizualizace struktury webu koncept zobrazuje i různé interaktivní prvky. Jedná se zejména o možnost přiblížení či oddálení vizualizace, což napomáhá v případě zkoumání rozsáhlejšího grafu a rovněž zrakově hendikepovaným uživatelům. Jelikož rozložení grafu nemusí být vždy efektivní, je umožněno uživateli vytvořit vlastní rozložení uzlů jejich přetažením a následným ukotvením na dané pozici. Za další interaktivní prvek by se dalo považovat kontextové menu, které je zobrazeno až při kliknutí na konkrétní uzel. Cílem kontextového menu je dát uživateli možnost zobrazit detailnější informace o konkrétním uzlu. Koncept tuto situaci zachycuje spolu s informačním panelem obsahujícím další vizualizace vygenerované např. z externího zdroje jako je Google Ana-



Obrázek 21: Koncept vizualizace struktury webu s prvky interakce

lytics. Informační panel je zobrazen až s výběrem konkrétní položky v menu uzlu. Menu nabídka může být rozšířena o další položky a nabízet tak ještě podrobnější informace, např. z dalších externích zdrojů.

Jelikož struktura webu může obsahovat velké množství uzlů, je vhodné koncept vizualizačního nástroje doplnit o různé typy filtrů. Filtry mohou být zaměřené jednak na dostupná data s cílem vizualizovat pouze určitou část struktury webu, o kterou má uživatel zájem a zároveň na samotný graf. Filtry zaměřené na data mohou pomoci uživateli najít, např. s použitím regulárního výrazu, uzly se specifickým tvarem URL adresy, případně uzly obsahující klíčové slovo třeba v dostupných meta atributech. Filtry zaměřené na samotný graf mohou uživateli pomoct identifikovat různé situace, které v grafu mohou nastat. Pro implementaci byly zvoleny následující filtry:

- vyhledávání podle URL adresy,
- vyhledávání podle titulku stránky,
- vyhledávání podle meta atributů,
- zobrazení všech uzlů, které mají výstupní hranu,
- zobrazení všech uzlů, které mají vstupní hranu,
- zobrazení uzlů, které odkazují sami na sebe,
- zobrazení dvojce vzájemně propojených uzlů.

Kromě zmíněných filtrů lze využít i jiných metod, např. shlukování uzlů. Sloučení více uzlů do skupin podle různých kritérií, může výrazně zvýšit čitelnost a přehlednost grafu. Příkladem zmíněných kritérií může být např. velikost uzlu, HTTP status stránky nebo záznam z meta tagu *robots*.

4.3.4 Problematické faktory vizualizace

Vizualizace pomocí grafu dává uživateli obrázek o celkové struktuře webu. S narůstajícím počtem uzlů a hran však může snadno dojít k tomu, že se vizualizace stane nepřehlednou a nečitelnou, viz obr. 22. Tento případ může nastat u větších webů, jako je např. internetový obchod. Podle společnosti Cambridge Intelligence[19] existují čtyři problémy, které mohou nastat v případě vizualizace velkých sítí:

1. **Omezené množství pixelů na obrazovce** - rozlišovací schopnost monitorů je limitována, na obrazovce lze tedy zobrazit jen omezený počet uzlů určité velikosti.
2. **Omezený výpočetní výkon počítače** - při vizualizaci velkých sítí lze narazit na limity dostupného hardware, což může mít vliv na plynulost aplikace.
3. **Omezená kognitivní kapacita uživatelů** - lidský mozek není dokonalý a dokáže interpretovat nanejvýš několik set uzlů v jednom grafu. Při zkoumání detailů se toto množství redukuje na několik desítek uzlů.
4. **Exponenciální nárůst hran se zvyšujícím se počtem uzlů** - velká provázanost v rámci datového souboru, může mít za následek hustou koncentraci vzájemně propojených uzlů v jednom místě.

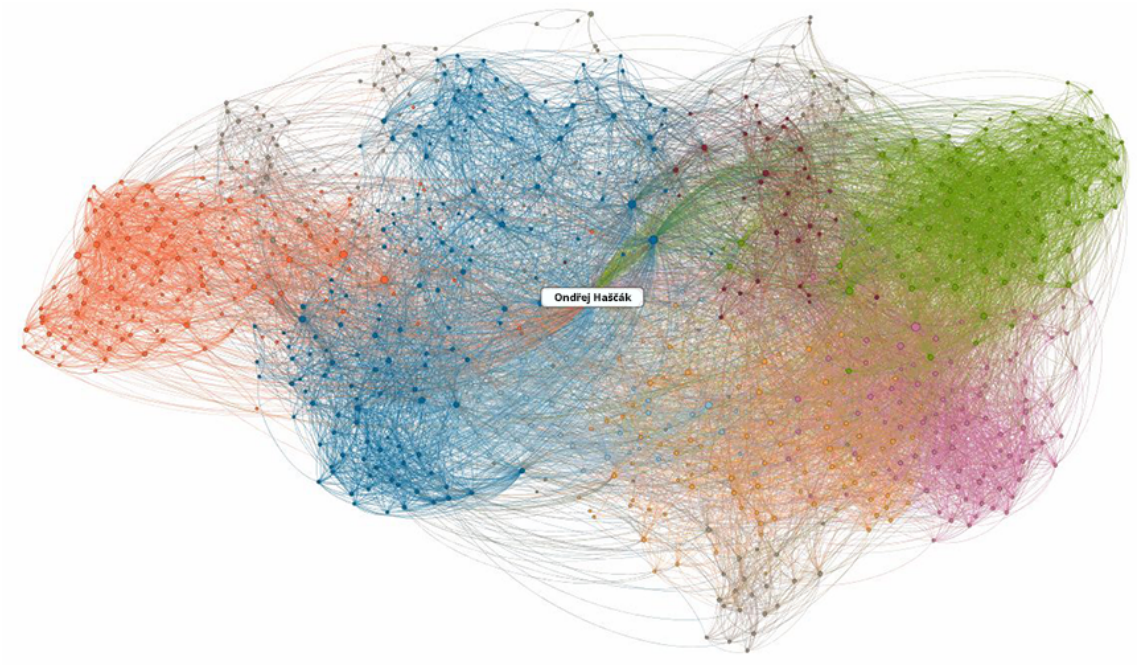
Při řešení tohoto problému lze v případě rozsáhlého grafu přistoupit k různým technikám. Jednou z nich je redukce hustoty síťového grafu. K tomu lze využít následující přístupy:

- **Seskupení uzlů** - uzly mohou být seskupeny na základě společných atributů. Seskupené uzly pak mohu být na vyžádání rozšířeny tak, aby uživateli zobrazily vztahy mezi jednotlivými uzly.
- **Sloučení uzlů** - často může dojít k duplicitě dat v případě, že se pracuje s různými datovými zdroji. Spojením duplicitních uzlů do jednoho umožňuje uživatelům pracovat na zvolené úrovni detailu.

Alternativní technikou je nezobrazovat uživateli najednou celý graf, ale jen určitou jeho část. V tomto případě lze využít následujících přístupů:

- **Filtrování** - v případě, že se uživatel chce zaměřit jen na určité detaily v datovém souboru jsou filtry ideálním prostředkem jak rozštěpit velký graf na menší tak, aby obsahoval jen pro něj klíčové informace.
- **Rozšíření na požádání** - uživateli je na začátku zobrazen jeden uzel nebo malá síť, kterou lze inkrementálně rozšiřovat na vyžádání. Uživateli je tak nabídnuta možnost přehledným způsobem prozkoumat celý graf nebo jen jeho určitou část.

- **Měření centrality** - měření centrality umožňuje identifikovat významné uzly v rámci sítě. Například uzly, které mají nejvíce spojení nebo uzly ležící nejčastěji na nejkratších cestách mezi ostatními uzly. S pomocí těchto informací lze identifikovat a vizualizovat spojení nebo podsítě, jež budou uživatele s největší pravděpodobností zajímat.



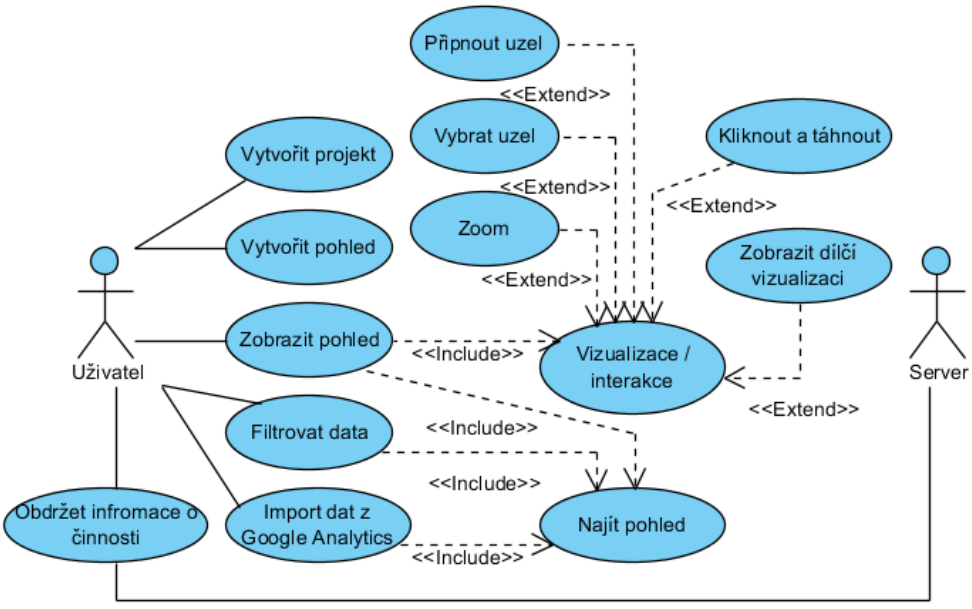
Obrázek 22: Vizualizace profesní sítě LinkedIn

5 Návrh a implementace

Hlavním cílem této kapitoly je implementovat navržený koncept vizualizace a sběru dat s ohledem na existující vizualizační knihovny a možnosti současných webových technologií.

5.1 Specifikace požadavků

Aplikace by měla uživateli umožnit vizualizovat strukturu webových prezentací dle navrženého konceptu do určité předem zvolené hloubky. Struktura webu bude zaznamenána jako pohled na web, který byl vytvořen v určitém čase. Aby uživatel mohl sledovat vývoj struktury, bude mu umožněno vytvářet více pohledů u konkrétního webu. Starší pohledy mu budou však stále k dispozici. Uživateli bude, pro potřeby hlubší analýzy, umožněno crawlerem získaná data doplnit daty o návštěvnosti z analytického nástroje Google Analytics. Jelikož činnost crawlera může trvat delší dobu, bude umožněno jeho činnost kdykoliv ukončit bez ztráty již získaných informací o struktuře webu. V případě předčasného ukončení bude uživateli zobrazena vizualizace pouze prohledané části struktury webu. V průběhu výkonu crawlera bude uživatel průběžně informován o jeho aktuálním stavu činnosti. Výsledná vizualizace bude implementována s ohledem na navržený koncept interakce s uživatelem a umožní mu filtrovat data tak, jak je popsáno v kapitole 4.3. Celkový pohled na systém z uživatelského hlediska je zobrazen diagramem případů užití na obr. 23.

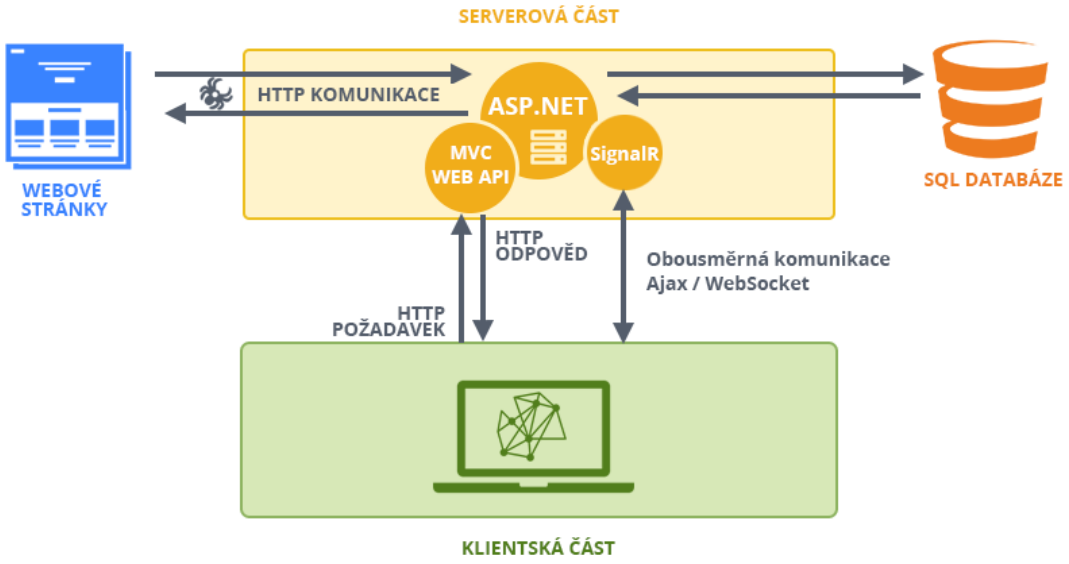


Obrázek 23: Diagram případů užití

5.2 Návrh řešení

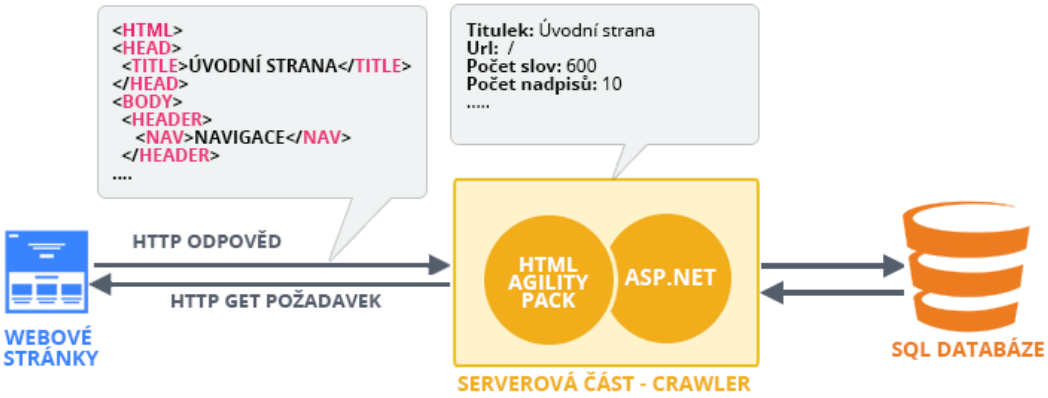
Aplikace je rozdělena na klientskou a serverovou část. Na straně klienta jsou v rámci webového prohlížeče využity technologie HTML 5, CSS 3 a JavaScript. S ohledem na různé typy a verze prohlížečů je aplikace optimalizována a vyvíjena pouze na prohlížečích Opera a Chrome v aktuální verzi. Hlavním cílem klientské části je dynamická prezentace obsahu uživateli a zároveň zprostředkování komunikace mezi uživatelem a serverovou částí aplikace.

Na straně serveru jsou využity primárně technologie společnosti Microsoft. Jedná se zejména o frameworky ASP.NET MVC, ASP.NET Web API, které jsou společností distribuovány pod Open Source licencí. Severovou část aplikace doplňuje databáze běžící na platformě MS SQL serveru. Hlavním úkolem serverové části je zprostředkovávat komunikaci s klientskou částí aplikace, komunikovat s databází a zpracovávat data nasbíraná crawlerem. Základní struktura navržené aplikace je zobrazena na obr. 24.



Obrázek 24: Struktura aplikace

Činnost crawlera je řízena serverem. Při prohledávání crawler vystupuje v roli prohlížeče, který simuluje procházení webu s pomocí HTTP protokolu. Získaná data jsou na serveru průběžně zpracovávána a ukládána do databáze. Uložená data jsou z databáze poskytnuta na podnět z klientské části aplikace dotazem na existující webovou službu. Kromě HTTP protokolu je pro vzájemnou komunikaci využíván i WebSocket protokol, jenž ovšem nemusí být podporován ve starších prohlížečích. Podrobnější informace o použitých technologiích a principech jsou popsány v následující kapitole.



Obrázek 25: Struktura crawlera

5.2.1 Detaily řešení

Veškerá hlavní funkčnost aplikace se odehrává na straně serveru, jelikož data pro vizualizace je nejdříve nutno nasbírat, vhodným způsobem upravit a uložit. Základním stavebním kmenem aplikace je ASP.NET MVC 5 projekt v kombinaci s frameworkem ASP.NET Web API, který umožňuje tvorbu jednoduchých HTTP služeb. Aplikace je dále složena z dalších čtyř projektů, jejichž kombinace tvoří vícevrstvou architekturu. Vícevrstvá architektura projektu napomáhá k minimalizaci závislostí uvnitř systému, zvyšuje jednoduchost a zjednodušuje testovatelnost aplikace. Díky tomuto přístupu je do budoucna umožněno aplikaci déle efektivně rozvíjet nebo jednoduše zaměnit jednu vrstvu za druhou. S využitím návrhových vzorů Repository a UnitOfWork je v kombinaci s ORM Entity Frameworkem ve verzi 6.1.1 zajištěna komunikace s databází. Pro tvorbu databáze byl využit přístup *Code First*, který vytváří databázi z kódu na základě předem definovaných atributů entit a anotací. Komunikaci s klientem zajišťuje datový formát JSON. Pro podporu tohoto datového formátu na straně serveru bylo nutné pomocí *NUGET* balíčkového systému doinstalovat rozšíření *JSON.NET*[6], jehož autorem je *James Newton-King*. Posledním důležitým prvkem architektury aplikace je návrhový vzor *Inversion of Control* (IoC)[7], jehož implementaci v projektu zajišťuje baliček *Autofac*[8] ve verzi 3.5.2. Podstatou tohoto návrhového vzoru je snížení závislostí mezi jednotlivými částmi systému, zavedením tzv. IoC kontejneru, který se stará o *dependency injection* (DI) neboli vkládání závislostí.

Aplikace na straně klienta využívá open-source JavaScriptový framework AngularJS vyvinutý společností Google. Framework je založen na návrhovém vzoru MVC, čímž umožňuje vytvářet dobře strukturované, jednoduše testovatelné a udržovatelné

[6]Dokumentace a zdrojové kódy jsou k dipozici na stránce http://www.newtonsoft.com/

[7]Více informací o tomto návrhovém vzoru je k dispozici na stránce Martina Fowlera http://martinfowler.com/articles/injection.html

[8]Dokumentace a zdrojové kódy jsou k dipozici na stránce http://autofac.org

font-end aplikace. Ačkoliv AngularJS obsahuje mnoho zajímavých funkcí, byl doplněn o několik rozšiřujících modulů. Mezi podstatné patří tyto následující:

- **UI-Router** - modul nahrazující vestavěný *ngRoute* modul. Stejně jako původní modul zajišťuje směřování URL adres na odpovídající šablony, avšak založené na stavech. Konkrétnímu stavu pak odpovídá jedna URL adresa, controller a šablona. Zároveň lze stavy a pohledy uspořádat do hierarchie, což se může hodit v případě tvorby komplexnějšího uživatelského rozhraní aplikace.
- **Restangular** - je AngularJS služba, která značně zjednodušuje dotazování pomocí základních HTTP metod GET, POST, DELETE a UPDATE. Hodí se zejména u aplikací komunikujících se serverem pomocí RESTful API.

Díky nutnosti průběžně informovat uživatele o průběhu činnosti crawlera se na straně klienta i serveru využívá knihovna SignalR[9]. Knihovna zajišťuje vzájemnou obousměrnou komunikaci mezi klientem a serverem v reálném čase. K tomu využívá sadu různých podpůrných technologií jako je např. HTML 5 WebSockets, Ajax aj. Díky možnosti obousměrné komunikace je knihovna rovněž využita v případě nutnosti zastavení činnosti crawlera samotným uživatelem.

5.2.2 Implementace automatizovaného crawlera

Jedním z hlavních úkolů crawlera, jak již bylo popsáno v kapitole 4, je identifikovat odkazy v rámci konkrétní webové stránky. Schopnost crawlera odkazy na stránce správně identifikovat se odvíjí od jeho znalosti jazyka HTML, respektive specifických elementů tohoto jazyka a jejich atributů. Aby se crawler lépe orientoval v HTML kódu, byla pomocí balíčkového systému doinstalována knihovna *Html Agility Pack*[10]. Hlavní předností této knihovny je integrovaná podpora jazyka XPath, s jehož pomocí lze velmi efektivně vyhledat požadované elementy v rámci DOM. Detekovat všechny odkazy v rámci stránky, včetně dalších informací, které se vážou k jiným elementům, je pak otázkou pár řádků kódu viz výpis č.1.

```
1 //Nalezení elementu 'description'
2 HtmlNode nodeDesc = doc.DocumentNode.SelectSingleNode("//meta[@name='description']");
3 //Nalezení elementu 'title'
4 HtmlNode title = doc.DocumentNode.SelectSingleNode("//title");
5 //Nalezení všech odkazů na stránce obsahující atribut 'href' a definované elementem 'a'
6 var linkNodes = doc.DocumentNode.SelectNodes("//a[@href]");
```

Výpis 1: Ukázka použití knihovny Html Agility Pack k nalezení HTML elementů

Ještě před samotným prohledáváním struktury stránky je nutné stránku z internetu stáhnout a načíst do paměti. Tuto činnost zajišťují třídy .NET frameworku *Htt-*

[9] Knihovna je dostupná zdarma ke stažení na adrese http://signalr.net
[10] Dokumentace a zdrojové kódy knihovny na adrese http://htmlagilitypack.codeplex.com/

pWebRequest a *HttpWebResponse*. Jakmile jsou získány všechny dostupné hledané informace, crawler si uloží informaci o navštíveném odkazu a dále přidá do fronty zpracování všechny odkazy, zatím nenavštívené. Crawler definuje událost *PageAdded*, která je vyvolána při každé návštěvě. Událost předává informace o přidané stránce svým předplatitelům jako je třída *SpiderController*. Jakmile je událost vyvolána, spustí se obslužná metoda, která prostřednictvím knihovny SignalR zašle asynchronně zprávu klientovi. Tímto je zajištěna potřeba průběžně informovat uživatele o průběhu činnosti crawlera. Jelikož činnost crawlera probíhá v cyklu a může trvat neznámou dobu, je dobré mít možnost ho nějakým způsobem zastavit. První možnost jak zastavit crawlera je definovaná uživatelem před jeho samotným spuštěním, a to parametrem tzv. hloubky zanoření. Uživatel si zvolí číslo, které určuje hloubku, do níž se crawler zanoří. Tato hloubka se zvyšuje pokaždé, když crawler z dané stránky vyexportuje nenavštívené odkazy. Pokud se hloubka rovná zvolenému číslu, činnost crawlera je ukončena. To ovšem může taky nějakou dobu trvat, proto byla implementována možnost zastavit crawlera okamžitě rovněž pomocí knihovny SignalR. Pro integraci této knihovny na straně klienta byla vytvořena AngularJS služba *signalRSvc*. Zároveň byl soubor s šablonou *LayoutAdmin.cshtml* doplněn o referenci na knihovnu a automaticky, severem vygenerovaný, *hub* proxy skript. Vygenerovaný proxy skript ulehčuje navázat komunikaci se serverem a zároveň psát obslužné metody, které může volat server na klientovi, či metody umožnující volat funkce na serveru. Vytvořená služba zprostředkovávající komunikaci v rámci obou stran je zobrazena ve výpise č.2.

```
1  app.service('signalRSvc', function ($, $rootScope) {
2      var proxy = null;
3
4      // Metoda volající funkci serveru, která zastaví činnost crawlera
5      var stopProcessing = function() {
6      // Navázání spojení se serverem. Jelikož je využíváno dynamického generování proxy skriptů,
         není třeba manuálně vytvářet objekt connection
7          var job = $.connection.pageHub;
8          // Volání metody na serveru, která spustí událost ukončující činnost crawlera
9          job.server.stopWork();
10     }
11
12     var initialize = function () {
13         var job = $.connection.pageHub;
14         // Vytvoření obslužné metody, kterou volá server, když informuje o činnosti crawlera
15         job.client.addNewMessageToPage = function (object, message) {
16             // Propagace události napříč aplikací
17             $rootScope.$emit("addNewMessageToPage", object);
18         };
19         $.connection.hub.start(); // Vytvoří spojení mezi klientem a serverem
20     };
21
22     return {
23         initialize: initialize, stopProcessing: stopProcessing};
24 });
```

Výpis 2: AngularJS služba pro vzájemnou komunikaci mezi klientem a serverem

Třída *Spider.cs* implementuje výše popisovanou funkčnost crawlera a zároveň obsahuje reference na služby umožňující ukládat nasbíraná data do databáze. V rámci každé iterace je před vyvoláním události *PageAdded* ještě vytvořen objekt *Page*. Ten je po doplnění nasbíraných informací uložen do databáze právě pomocí služby *pageService*. Třída dále obsahuje funkce pro vyhodnocení počtu slov na stránce, které se provádí až po vyčištění zdrojového kódu stránky na čistý text. Kromě počtu slov lze ještě zjistit počet znaků bez mezer i s mezerami. Funkčnost třídy lze samozřejmě dále rozšiřovat a doplňovat o další zajímavé analýzy nad samostatnou stránkou. Crawler může být taky upraven pro zcela jiný účel, např. pro hledání specifických stran, nejen v rámci domény jednoho konkrétního webu.

5.2.3 Volba vizualizačního nástroje

Jak už bylo zmíněno v kapitole 3, aktuálně existuje mnoho způsobů, jak vizualizovat data na webu. Jelikož je již známá předběžná struktura dat a koncept vizualizace, zbývá vybrat vhodný nástroj, respektive knihovnu, která bude nejlépe vyhovovat pro účely této aplikace i pro její budoucí vývoj. Ze značného množství knihoven byly vybrány tři, které vyhovovaly předpokladům interaktivní vizualizace grafu. Jedná se o knihovny D3.js[11], Cytoscape[12] a Sigmajs[13]. Ačkoliv všechny knihovny umožňují vizualizovat strukturu webu, pro implementaci byla vybrána knihovna D3.JS. Oproti zmíněným knihovnám, které používají pro vizualizaci HTML 5 *Canvas*, využívá značkovací jazyk SVG. I když vykreslování křivek pomocí SVG může být pomalejší něž práce s rastrovanou grafikou v Canvasu, je použití tohoto přístupu výhodné především z interaktivní stránky. Další nespornou výhodou této knihovny je možnost využívat a kombinovat široké spektrum různých vizualizačních technik, nebo taktéž existence široké komunity vývojářů a rozsáhlé dokumentace v mnoha jazycích. Výpočty, prováděné knihovnou, jdou mimo jiné využít i pro kreslení na Canvas, čímž se dá využití jazyka SVG pro vizualizaci obejít. Vzhledem k silně interaktivnímu přístupu k vizualizaci, který má výsledná aplikace poskytnout, byla knihovna D3.JS vybrána jako vhodný nástroj i s ohledem na budoucí vývoj. Na rozdíl od většiny vizualizačních knihoven není D3.JS pouze předpřipravenou kolekcí vizualizací a widgetů. Přestože umožňuje vytvářet základní vizualizace, jako je třeba koláčový,síťový či sloupcový graf, její hlavní síla tkví ve flexibilitě a možnosti mít plnou kontrolu nad finálním výstupem. Typickým příkladem může být nutnost rozšířit koláčový graf o něco, co vizualizační knihovna ještě neimplementuje, např. změnu pozadí u legendy. U jiných knihoven by toto mohlo znamenat hodně práce v chaotickém prostředí, ovšem s D3.JS je to otázkou doplnění jen pár řádků kódu. To samé platí v případě nutnosti vytvoření úplně nové vizualizace, což je díky API D3.JS opravdu jednoduché, přehledné a efektivní.

[11] Knihovna dostupná na adrese http://d3js.org
[12] Knihovna dostupná na adrese http://cytoscape.github.io
[13] Knihovna dostupná na adrese http://sigmajs.org

5.2.4 Implementace vizualizačního konceptu

AngularJS je framework obsahující mnoho zajímavých nástrojů, které mohou značně ulehčit a zpřehlednit vývoj aplikace. Jedním z těchto nástrojů jsou tzv. direktivy. Direktivy dokážou rozšířit základní syntaxi jazyka HTML o novou funkcionalitu s podpůrným JavaScriptovým kódem a vystupují v roli nových elementů, atributů, komentářů nebo CSS tříd. Framework samotný v základu obsahuje množství již implementovaných direktiv, které jsou popsány v jeho API dokumentaci [14].

Direktivy je dobré používat v případě, že se kód často opakuje. Je nutné manipulovat s HTML DOM nebo je potřeba zaobalit funkčnost nějaké externí JavaSriptové knihovny. Jelikož byla pro implementaci zvolena knihovna D3.JS, lze očekávat znovupoužitelnost komponenty a v rámci vizualizace se bude manipulovat HTML DOM. V tomto případě je použití direktiv zcela na místě. Výpis č.3 představuje základní šablonu direktivy pro implementaci navrženého vizualizačního konceptu.

```
1  app.directive('force', ['$compile', function ($compile) {
2
3      return {
4          // definice typu direktivy (E – element, A – atribut, C – třída, M – komentář)
5          restrict : 'A',
6          // kopírovat obsah původního HTML ano / ne
7          transclude: true,
8          // manipulace s DOM
9          link: function (scope, element, attrs) {
10             // zde funkce pro manipulaci s DOM
11         }
12     }
13 }]);
```

Výpis 3: AngularJS služba pro vzájemnou komunikaci mezi klientem a serverem

Vytvořením této jednoduché direktivy vznikl nový atribut jazyka HTML s názvem *force*. Tento atribut lze uplatnit na jakýkoli element a jeho použitím se vyvolá kód direktivy, jenž bude v tomto případě převážně umístěn v rámci funkce *link*. Tato funkce obsahuje tři atributy: *scope*, *element* a *attrs*. První z nich představuje kontext, v němž je uložen model tak, aby k němu měly přistup controllery, direktivy, ale i pohledy pomocí tzv. expressions. Jelikož bylo v rámci direktivy explicitně zakázáno vytvořit nový kontext, byl automaticky poděděn z nejbližšího, který již v rámci hierarchie kontextů existuje. Tímto lze v rámci direktivy přistupovat k proměnným, které jsou vytvořeny v controlleru *SpiderDetailViewStrctureCtrl*, jenž řídí hlavní činnost vizualizačního nástroje. Parametr *element* představuje element, na němž je direktiva aplikována a je zabalen v objektu jQLite[15]. Poslední parametr *attrs* obsahuje normalizované atributy spojené s elementem, kterým je direktiva definována. Pomocí dodatečných atributů lze direktivu svázat s daty nebo ovlivnit její konfiguraci.

[14] K dipozici na adrese https://docs.angularjs.org/api/

[15] JQLite je odnož populární knihovny jQuery, která implementuje pouze její nejčastěji používané funkce s ohledem na malou velikost knihovny.

Jakmile uživatel vstoupí na stránku s vizualizací, je v rámci controlleru vyvolána metoda *graphData*. Metoda s pomocí knihovny Restagnular zašle HTTP GET požadavek na ASP.NET WEB API službu, která po zpracování dotazu vrací data získaná crawlerem ve formátu JSON. Jakmile jsou data k dispozici, jsou uložena do objektu *$scope.datas* a dále předána do direktivy jako hodnota atributu *datas*. Jelikož při vytváření direktivy ještě nemusí být data k dipozici, je v direktivě využita speciální metoda *$watch* sledující změny objektů ve *scope*. V případě, že jsou data k dipozici, následuje jejich zpracování a vizualizace pomocí knihovny D3.JS v rámci funkce *link*.

Knihovna D3.JS obsahuje množství již připravených vizualizací. Jedna z nich implementuje tzv. *force-directed layout*, s jehož pomocí lze modelovat síťový graf. Jak z názvu vyplývá, jedná se o typ grafového algoritmu, jenž k uspořádání jednotlivých prvků v prostoru využívá fyzikální simulaci sil. Implementace síťového grafu s využitím zmíněného layoutu tvoří základ pro vizualizační komponentu. Inicializaci layoutu grafu ukazuje výpis č.4. Na výpise je zobrazena pouze základní konfigurace ukončená metodou *start*, která spouští samotnou simulaci. V rámci konfigurace jsou nastaveny základní povinné parametry a předána potřebná data pro tvorbu uzlů a hran. Jelikož výsledné rozložení nemusí být vždy optimální, může být konfigurace doplněna o řadu parametrů pro jeho přizpůsobení. Výčet všech možností je popsán v dokumentaci[16]. V aplikaci byli použity pouze rozšiřující parametry *linkDistance* a *charge*. První parametr určuje délku hran mezi propojenými uzly, kde výchozí hodnota je 20. Druhý parametr je tzv. náboj. Jeho hodnota může být kladná i záporná. Čím více je hodnota záporná, tím více jsou uzly mezi sebou více odpuzovány a naopak.

```
1 var force = d3.layout.force()
2                     .size([width, height])   // rozlišení grafu
3                     .nodes(nodes)            // data pro uzly
4                     .links(edges)            // data pro hrany
5                     .charge(-500)            // náboj mezi uzly
6                     .linkDistance(180);      // délka hran mezi uzly
7                     .start();                // spuštění simulace
```

Výpis 4: Inicializace síťového grafu

Aby bylo vše funkční a vizualizace něco zobrazila, je nutné pro každý uzel a hranu vytvořit odpovídající objekty v SVG. Knihovna D3.JS svému *fluent* API tvorbu těchto objektů zjednodušuje viz výpis č.5. Kromě již zmíněného je v poslední řadě nastavena reakce na událost *tick*, která je vyvolána v každém kroku simulace, dokud není parametr *alpha*[17] blízký nebo roven nule. V rámci reakce na tuto událost jsou nastaveny hodnoty souřadnic uzlů a hran tak, aby respektovaly nastavení definované při počáteční inicializaci layoutu. Dále je zde specifikován tvar křivky tvořící hranu mezi jednotlivými uzly, a to i v případě, kdy uzel odkazuje sám na sebe. Vizualizace je dále rozšířena o různé interaktivní prvky, filtry a další doplňkové vizualizace, které se vážou na takto vytvořený síťový graf pomocí D3.JS knihovny.

[16] K dispozici na adrese https://github.com/mbostock/d3/wiki/Force-Layout

[17] Parametr tzv. ochlazení v rozsahu 0 až 1. Jeho hodnota vyjadřuje, jak probíhající simulace konverguje k vytvoření optimálního layoutu grafu.

```
// Vytvoření hran mezi uzly
var link = svg.append("svg:g").selectAll("link")
                              .data(edges)
                              .enter().append("svg:path")
                              .attr("class", "link");
// Vytvoření uzlů
var node = svg.append('svg:g').selectAll("node")
                              .data(nodes)
                              .enter().append("svg:g")
                              .attr("class", "node")
```

Výpis 5: Vytvoření uzlů a hran

5.2.4.1 Kontextová nabídka u uzlu Pro vytvoření kontextové nabídky u jednotlivých uzlů, tak jak je zobrazena na vytvořeném konceptu, je využit koláčový graf. Ten je následně upraven do podoby tzv. koblihového grafu. Jako základ pro vizualizaci slouží proměnná *menuDataSet*, která obsahuje jednotlivé položky menu ve formátu JSON. Položky menu jsou popsány atributy jako popisek, velikost nebo text ve formátu unicode pro zobrazení ikonky. Následně je pomocí metody *d3.layout.pie()* vytvořena funkce *pie()*, jíž je jako parametr předána proměnná *menuDataSet*. Knihovna se pak postará o transformaci JSON pole na objekty obsahující parametry *startAngle* a *endAngle* popisující jednotlivé výseče koláčového grafu. Ke konstrukci grafu jsou pak využity metody D3.JS pro tvorbu SVG objektů. Přeměna koláčového grafu na koblihový a vykreslení jednotlivých výsečí je demonstrována ve výpise č.6. Kontextová nabídka je následně připojena ke každému uzlu a spouštěna pomocí události *onClick*. Reakce na tuto událost je přidána jako další funkce ve výpise č.5 v sekci *vytvoření uzlů*.

```
// Vytvoření koblihového grafu nastavením vnitřního a vnějšího úhlu
var arc = d3.svg.arc().innerRadius(innerRad).outerRadius(outerRad);
// Vytvoření skupin pomocí elementu 'g' pro každý klín
var g = svgMenu.selectAll(".arc").data(pie(menuDataSet)).enter().append("g").attr("class", "arc");
// Vytvořený cest a obarvení klínů
g.append("path").attr("d", arc).attr(" fill ", function (d,i) {return color(i);});
```

Výpis 6: Vytvoření kontextové nabídky pomocí koláčového grafu

V rámci kontextové nabídky lze spustit informační panel. Ten, pokud jsou k dipozici data z nástroje Google Analytics, zobrazuje různé metriky vztahující se k dané stránce. Pro tvorbu tohoto panelu je využit velmi podobný postup jako u kontextové nabídky. Obsahuje ovšem speciální horizontální sloupcový graf zobrazující průměrnou dobu strávenou uživateli na stránce k době strávené na celém webu. Tento graf je další ilustrací toho, že v rámci knihovny D3.JS lze jednoduše a efektivně vytvářet vlastní vizualizace. Jelikož se zde pracuje s SVG, lze na jednotlivé prvky grafů navázat různé události. Ty se dají třeba využít k vytvoření tzv. *ToolTipu*. ToolTipy mohou výrazně ulehčit čtení hodnot z grafu a v tomto případě i z vytvořeného menu, které dle konceptu má být ikonové. Vytvoření ToolTipu je provedeno na začátku metody *link* a to přidáním elementu

div s třídou *tooltip* za element *body*. Pomocí D3.JS API je pak na daný prvek grafu, např. na zmiňovaný půlkruh, navázána událost *mouseMove*, jejíž implementace je zobrazena ve výpise č.7. Podobný postup je využit i v jiných případech využívajících tento interaktivní prvek.

```
1 .on("mousemove", function(d) {
2     div.style("display", "none");
3     div
4         .html(d.data.name + " <br /> Shlédnutí " + d.value) // Text obsažen v rámci ToolTipu
5         .style("left", (d3.event.pageX + 12) + "px") // Absolutní pozice zleva na obrazovce
6         .style("top", (d3.event.pageY - 10) + "px") // Absolutní pozice shora na obrazovce
7         .style("opacity", 1) // Zviditelnění
8         .style("display", "block");
9 })
```

Výpis 7: Implementace události pro zobrazení ToolTipu

5.2.4.2 Další interaktivní prvky Vizualizace obsahuje mnoho dalších interaktivních prvků. Jedním z nich je možnost připínat nebo odepínat uzly v rámci síťového grafu kdekoli na obrazovce. Uživatel si tak může zobrazenou strukturu přizpůsobit dle jeho potřeb. S přepínáním uzlů souvisí i možnost s uzly manipulovat a přetahovat různě po obrazovce. Podobně jako u *ToolTipů* je zde implementována reakce na událost, která se váže ke specifickému SVG elementu. V tomto případě existují události tři: *dragStrat*, *drag* a *dragEnd*. Element je zafixován již v události *dragStrat*, jež je vyvolána při stisknutí uzlu. Zároveň je mu nastavena vlastnost *dragging* na *true*. Následuje vyvolání události *drag*. Ta v průběhu pohybu myší snímá pozici kurzoru. Aktuální souřadnice předává SVG elementu, čímž mění jeho pozici. Poslední událost *dragEnd* je vyvolána na konci pohybu, kde je elementu nastavena vlastnost *dragging* na *false*. Uzel je na konci pohybu vždy ukotven. Pro jeho opětovné uvolnění je nutné provést dvojklik.

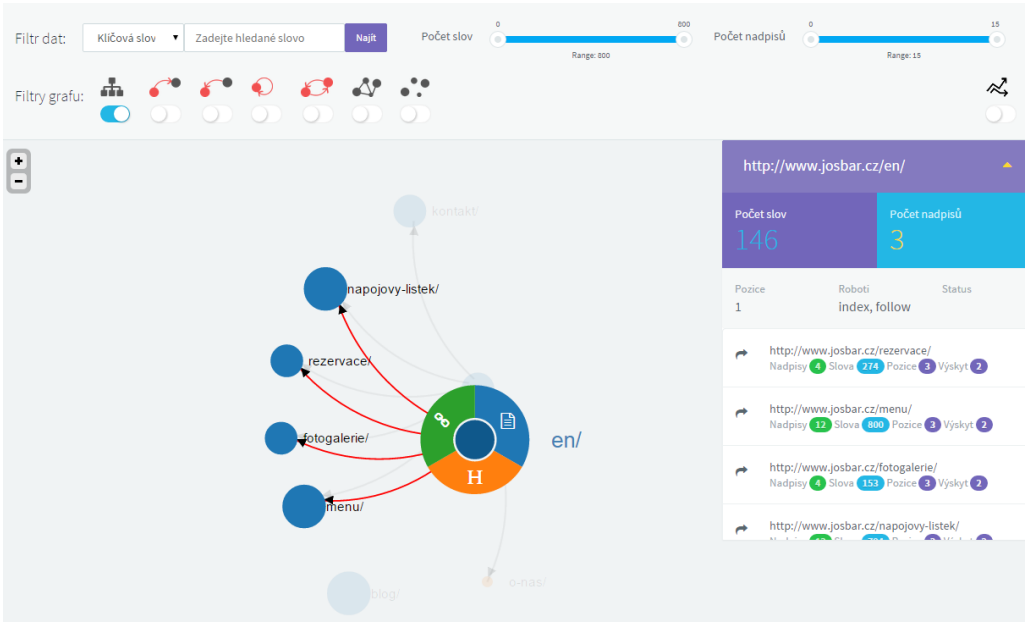
Za další implementovaný interaktivní prvek by se dala považovat možnost přiblížení či oddálení grafu. Knihovna D3.JS má tuto funkci již předpřipravenou v rámci objektu získaného voláním *d3.behavior.zoom()*. Na tento objekt je navázán listener na událost *zoom*, v němž je vyvolána metoda *onZoomChanged*. Součástí metody je kód, jenž je aplikován na obalující SVG kontejner pro celou vizualizaci. Obsahuje transformační SVG metody, které na základě události o aktuálním měřítku přiblíží či oddálí pracovní plochu. Toto chování lze vyvolat pohybem kolečka myši, ale bylo rovněž implementováno manuální ovládání pomocí vizuálních tlačítek.

Poslední prvek, který stojí za zmínku, je postranní panel, v němž jsou zobrazeny podrobnější informace o uzlu. Tento panel nebyl zahrnut v rámci navrženého konceptu, ale byl do nástroje začleněn za účelem více zpřehlednit dostupné informace o uzlu. Panel se zobrazí spolu s popisovanou kontextovou nabídkou při kliknutí na uzel. Obsahuje přehled všech vstupních a výstupních hran, respektive odkazů, které se na stránce vyskytují nebo na ni vedou. U každého odkazu dodatečně uvádí pozici, na níž byl nalezen v rámci stránky a další alternativní pozice, pokud se odkaz vyskytoval na více místech.

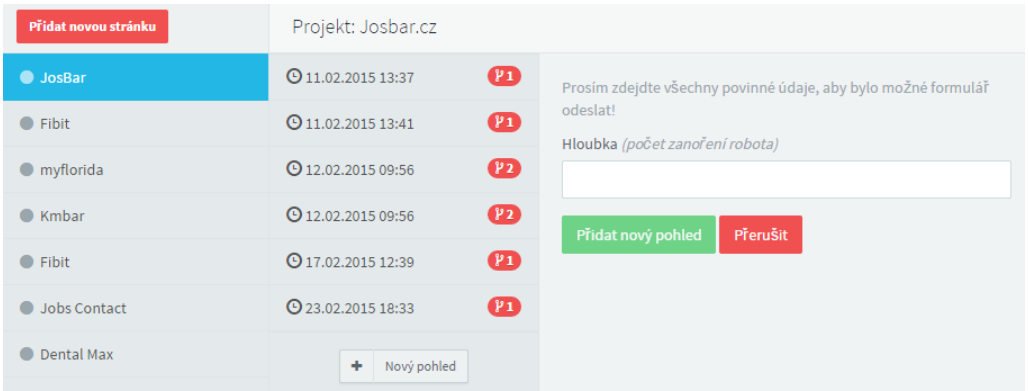
Dále jsou u odkazů specifikovány informace o počtu slov a nadpisů tak, aby uživatel dostal úplný obrázek o důležitosti odkazu.

5.3 Výsledná aplikace

Podoba výsledné aplikace je zobrazena na obrázcích 26 a 26. Obrázek 26 ilustruje pracovní prostředí vizualizačního nástroje s aktivovanou kontextovou nabídkou u uzlu spolu s postranním a informačním panelem. Na druhém obrázku je zobrazeno prostředí pro vytvoření nové vizualizace respektive pohledu.



Obrázek 26: Ukázka výsledné aplikace - vizualizace



Obrázek 27: Ukázka výsledné aplikace - vytváření pohledu

6 Případové studie

Tim Berners-Lee, považovaný za zakladatele dnešního internetu, na svém Twitter účtu v roce 2014 oznámil, že byl překročen milník počtu aktivních webových stránek na internetu. Ten v té době přesáhl, historicky poprvé, jednu miliardu. V dnešní době si může webovou stránku vytvořit téměř kdokoli, a to i za pár minut. S narůstajícím obsahem obvykle přibývá počet stránek, které, jak už bylo zmíněno v kapitole 4, se mohou seskupovat do různých struktur. Vytvořená aplikace pomáhá tyto struktury identifikovat a doplňuje je o další užitečné informace. Tato kapitola je zaměřena na využití vytvořené aplikace k vizualizaci webové prezentace s ohledem na její velikost.

6.1 Firemní prezentace společnosti Sunshine Real Estate s.r.o.

Sunshine Real Estate je poměrně mladá společnost založená v roce 2009. Zabývá se poskytováním různých služeb v oblasti investic do nemovitostí, a to především na území USA. Na svém webu společnost prezentuje jak sebe, tak nabízené nemovitosti a služby v různých jazycích. Ačkoli společnost vznikla před šesti lety, její současné webové stránky vznikly v roce 2014. [18] Jedná se tedy o poměrně nový web, což dokazuje i využití nových elementů jazyka HTML 5 v rámci zdrojového kódu stránky.

Tento typ webu se dá obecně charakterizovat jako malý web. Podobně jako osobní stránky, či menší blog, se skládá z řádově desítek stránek. Jeho obsah lze pomocí vytvořeného nástroje analyzovat a celkově vizualizovat bez nutnosti předčasně ukončit činnost crawlera. Jelikož cílem webových stránek je na určité úrovni prezentovat nabízené služby a osobnost společnosti, je žádoucí, aby při procházení působil web na návštěvníky dobrým dojmem. Z opačné stránky je zas nutné, aby se provozovatel o web staral, průběžně jej aktualizoval a vylepšoval, ať už na základě podnětů jeho návštěvníků, či různých statistik. Před spuštěním webu do produkčního módu mohly pak vzniknout následující otázky:

- Dostanou se uživatelé pomocí navigační struktury na všechny důležité stránky?
- Jsou všechny odkazy na webu funkční?
- Existuje stránka, která nemá žádný, nebo nekompletní obsah?

Odpovědět na tyto otázky by pro člověka, který má na starost tvorbu obsahu, vyžadovalo mnoho úsilí. V nejhorším scénáři by musel procházet a manuálně testovat každý odkaz na webu. Tato práce by byla časově náročná, nepřehledná a jelikož se člověk může lehce splést, nemusí být odvedena stoprocentně. V případě, že se vše podařilo manuálně zkontrolovat, nastala chvíle spuštění nové webové prezentace a mohly vniknout další otázky:

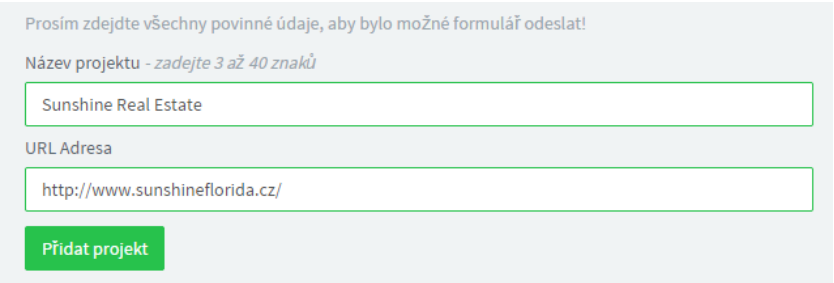
[18] Stáří webu bylo určeno pomocí nástroje na serveru http://web.archive.org/ a dnem první registrace domény.

- Stráví uživatelé na stránce s velkým obsahem více času?
- Má velikost obsahu vliv na návštěvnost?
- Je stránka s nejvyšším počtem nadpisů i nejnavštěvovanější?

Odpovědi na tyto otázky je nutné brát z různých analytických nástrojů, ovšem, často bez informací o vzájemných vazbách mezi stránkami. Informace generované analytickými nástroji spolu s informacemi o struktuře webu mohou být pro společnost klíčové v případě dalšího rozvoje prezentace. Kromě toho může, např. zpětná vazba od návštěvníků, vést k dalším úpravám webu. Proto je nutné brát v úvahu možnost vzniku nového obsahu i navigační struktury, která celý proces kontroly spouští od začátku. Podoba webu se tak v čase stále mění a bylo by dobré mít o těchto změnách i jejich důsledcích přehled.

6.1.1 Aplikace nástroje

K hledání odpovědí na kladené otázky byl využit vytvořený nástroj schopný analyzovat a vizualizovat obsah webu. Pro získání potřebných dat bylo nejdříve nutné vytvořit v rámci aplikace nový projekt. Vytvoření projektu je pouze otázkou správného zadání URL adresy stránky, na které crawler započne svou činnost.

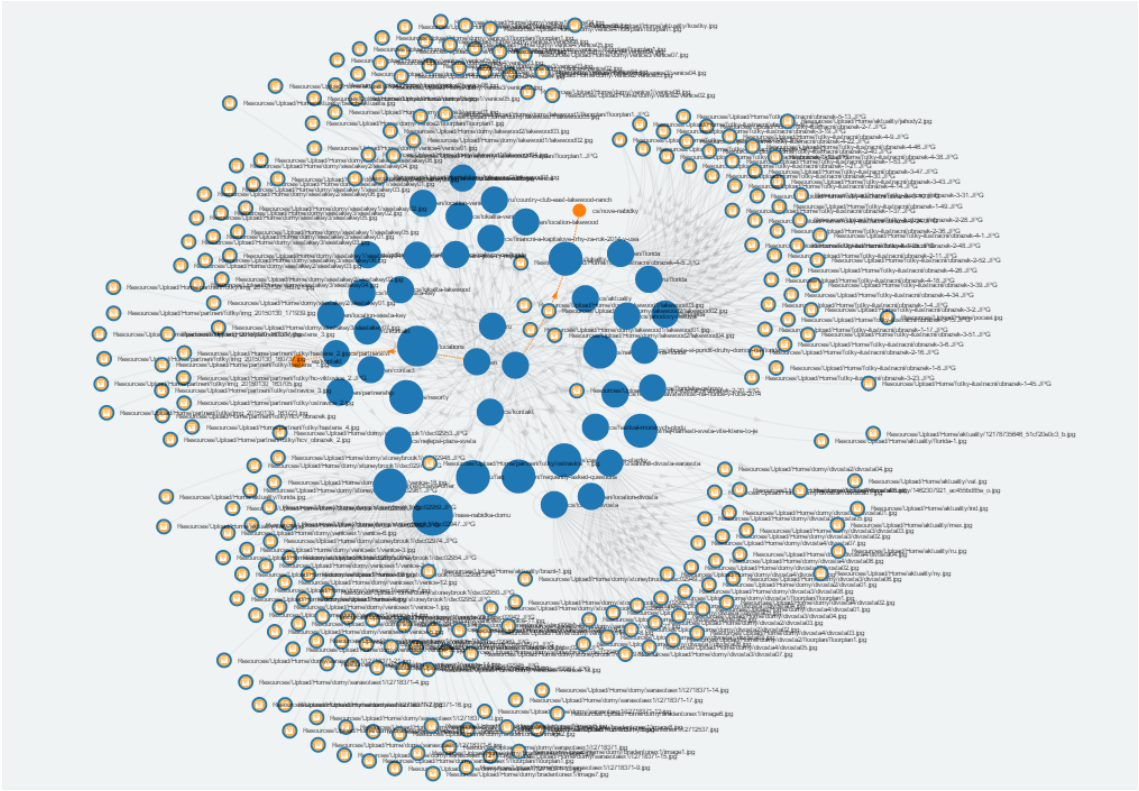


Obrázek 28: Vytvoření nového projektu

V dalším kroku byl vytvořen nový pohled reprezentující obraz webu v určitém čase. Při vytváření pohledu byla nastavena hloubka zanoření crawlera na číslo 10. Jelikož web není velmi rozsáhlý, bylo tímto zajištěno prohledání jeho kompletní struktury bez předčasného zastavení crawlera. Při průchodu webem bylo detekováno 291 uzlů, které propojovalo celkem 1158 hran. Z toho 246 uzlů tvořily obrázky a zbytek představoval jeho obsahovou část. Kromě těchto informací bylo zjištěno, že nejobsáhlejší stránka na webu čítá přibližně 1067 slov a existuje stránka obsahující až 20 nadpisů.

Odpovědi na kladené otázky byly následně hledány ve vizualizaci viz obr. 29, jež byla vytvořena na základě dat nasbíraných crawlerem. Ve výchozím stavu je pro přehlednost vizualizace ochuzena o odkazy - hrany, které se vyskytují v rámci navigace, či v patičce webu. Vyřazení těchto hran eliminovalo mnoho cyklů mezi uzly, jenž vznikají

v důsledku přítomnosti hlavní navigace a patičky na každé stránce prezentace společnosti. Nepřítomnost navigačních odkazů nevedla k situaci existence osamocených uzlů, lze tedy tvrdit, že každá stránka na webu je dosažitelná uživatelem.



Obrázek 29: Vizualizace obsahu webu společnosti Sunshine Real Estate

Ve vizualizaci na obr.29 lze rovněž identifikovat dva oranžové uzly, na než vede odkaz, ale sami osobě nikam dále neodkazují. Při analýze obou uzlů bylo zjištěno, že stránky vrací chybu *404* protokolu HTTP. Jedná se o chybu, která říká, že stránka pod danou URL adresou nebyla nalezena. Na webu tedy existují odkazy s jejichž pomocí se uživatel může dostat na neexistující stránky. Vizualizace tuto skutečnost zohledňuje barevným rozlišením uzlů a hran.

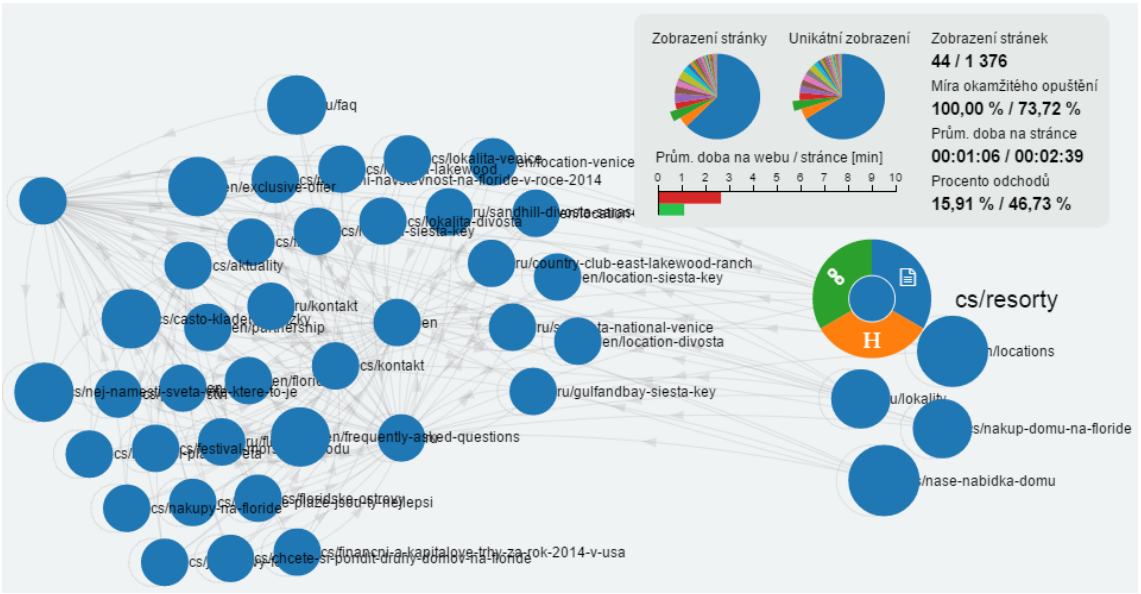
Pomocí filtru počtu slov, který nástroj integruje, byly identifikovány uzly, jenž mají malou koncentraci slov na stránce - (přibližně mezi 0 až 238). U těchto uzlů se dala předpokládat neúplnost obsahu, proto byly všechny stránky navštíveny a zkontrolovány. Výsledek kontroly je zobrazen v tabulce č.1. Opět zde byly poměrně jednoduše odhaleny nefunkční stránky s hláškou *Error 404* a to díky tomu, že crawler u stránek s chybovou hláškou nezkoumá jejich obsah. Počet slov je tedy u těchto stránek nulový.

Kromě dat o struktuře vygenerovaná crawlerem byla získána analytická data z nástroje Google Analytics ve formátu CSV. Importem získaných dat se podařilo vizualizaci obohatit o další informace vztahující se k jednotlivým uzlům. Metodou manuálního

Název uzlu	Počet slov	Kontrola
/cs/partnerstvi	216	OK
/en/partnership	233	OK
/cs/jahodovy-festival	227	OK
/cs/festival-morskych-plodu	233	OK
/cs/nove-nabidky	0	Error 404
/en/kontakt	0	Error 404

Tabulka 1: Možné filtry vizualizačního nástroje

připnutí uzlů na obrazovku byly odfiltrovány stránky s největším obsahem. Ten se pohyboval v rozmezí 941 až 1067 slov. Ostatní uzly nepřesahovaly hranici 690 slov. Z vizualizace následně vyplynulo, že jedna z nejobsáhlejších stránek patří mezi druhou nejvíce navštěvovanou stránku na celém webu a je v češtině. Ovšem mezi nejobsáhlejšími se vyskytovaly i stránky v jiných jazycích, jejichž návštěvnost byla téměř nulová. Fakt, že web hostuje na české doméně, mohl ovlivnit příchod zahraničních návštěvníků. Z výsledného pozorovaní tedy vyplynulo, že nejobsáhlejší stránky v české lokalizaci patřily mezi ty nejnavštěvovanější. Dále byly odfiltrovány stránky s nejvyšším počtem nadpisů. Mezi tyto stránky patřily úvodní strany webu ve všech lokalizacích. Bylo potvrzeno, že stránka s nejvyšším počtem nadpisů je nejvíce navštěvovaná v české a anglické lokalizaci, ovšem u ruské lokalizace tomu tak nebylo.



Obrázek 30: Ukotvení uzlů s největším počtem slov - detail s informačním panelem

6.1.2 Zhodnocení

V případové studii se podařilo vizualizovat obsah webové prezentace společnosti Sunshine Real Estate. Z výsledné vizualizace se jednoduše odhalil výskyt dvou nefunkčních odkazů na webu. V rámci webu nebyly nalezeny stránky, na něž by se bez přítomnosti hlavního menu nedalo dostat. Při analýze obsahu se povedlo identifikovat stránky s malým počtem slov. Mezi těmito stránkami se nevyskytovala ani jedna s nekompletním, či žádným obsahem. Díky dodaným datům z nástroje Google Anylytics bylo možné u jednotlivých uzlů zobrazit informační panel se statistikou návštěvnosti. V kombinaci s využitím existujících filtrů a interaktivních prvků, jako je např. možnost ukotvení uzlu, bylo umožněno uživateli zkoumat strukturu webu z různých pohledů. Informace získané pomocí vizualizačního nástroje by společnost mohla využít k odstranění nedostatků v rámci své prezentace nebo jako vstup pro další analýzu.

6.2 Informační portál ProŽeny.cz

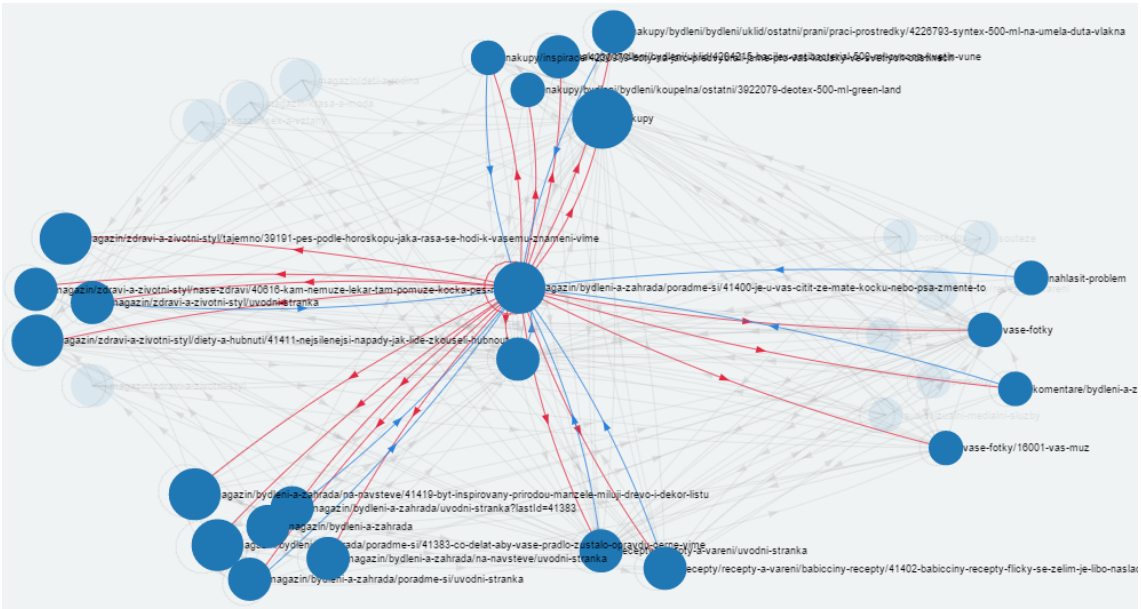
Informační portál představuje typ webu, na jehož rozvoji se pracuje každý den. Oproti předchozí případové studii zde počet stránek stoupá na tisíce až desetitisíce. Kromě velkých informačních portálů lze mezi weby s podobným charakterem zařadit i internetové obchody, či různé sociální sítě. Průchod crawlera při analýze tak rozsáhlého webu může trvat o mnoho déle, než v předchozím případě. V rámci vizualizačního nástroje lze rovněž předpokládat problém se zobrazením celkové struktury webu. Stránky českého online magazínu *prozeny.cz* jsou konkrétním příkladem takového webu. Tento magazín kromě publikace článků na svém webu nabízí, např. poradenství, různé soutěže, sekci s recepty a nákupy. Jelikož je web velmi obsáhlý a jeho charakter je jiný než v první případové studii, mohou při jeho provozu vznikat i jiné otázky:

- Na jaké články / stránky se dostanu z konkrétního článku?
- Jsou odkazované články ze stejné kategorie?
- Mají vzájemně odkazované články stejně velký obsah?
- Existují nějaké nefunkční odkazy, při procházení webu z dané pozice?

6.2.1 Aplikace nástroje

Stejně jako v předcházející případové studii je pro zodpovězení otázek využita vytvořená aplikace. Jelikož se jedná o jiný web a otázky se mohou týkat jeho různých částí, bylo nutné, vzhledem k povaze webu, vybrat vhodný vstupní bod pro crawlera. URL adresa, na níž započne prohledávání, musí být zvolena tak, aby crawler pokryl vybranou oblast zájmu. To samé platí při nastavení hloubky zanoření crawlera. Pro účel analýzy

byl vybrán konkrétní článek na webu ze sekce magazín - bydlení[19]. Aby bylo možné odpovědět na zmíněné otázky, byla nastavena URL adresa vybraného článku jako vstupní bod crawlera s hloubkou zanoření na jedna. Prohledaly se tedy pouze odkazy, které se nacházely na vstupní stránce a následně odkazy na nalezených stránkách. To vše v rámci domény online magazínu. Ve výsledku bylo nalezeno 33 uzlů, jenž propojovalo celkem 675 hran. Opět byl v nástroji zapnut filtr na vyřazení hran ve formě odkazů, které se nachází v navigační části a v patičce webu. Tímto krokem byl zredukován počet hran na 258, což je méně, než polovina. Výsledná vizualizace je na obr. 31.



Obrázek 31: Vizualizace vybrané části struktury webu online magazínu

Přeuspořádáním uzlů na obrázku podle sekcí, které šly rozpoznat z názvu uzlu (URL adresy), vnikla vizualizace, z níž lze mnohé vyčíst. Ve středu obrázku se nachází uzel reprezentující vstupní článek, pod ním se nachází uzel reprezentující hlavní stranu webu. Jak je z obrázku vidět, tak v rámci obsahu článku se lze dostat do různých kategorií webu i bez použití hlavního menu. Kromě toho lze v levé části pozorovat čtyři uzly obsahující články a stránky na témata ze stejné kategorie jako je vybraný článek. Dále se v dolní a horní části obrázku vyskytuje dalších třináct uzlů s články ze třech různých kategorií magazínu. Zbytek uzlů tvoří navigační a různé informační stránky, či kategorie webu. Zašedlé uzly a hrany představují v tomto případě stránky a odkazy, na něž se nedá v rámci obsahové části článku dostat. V případě vypnutí zmíněného filtru odkazů se na toto stránky dostat lze. Z vizualizace je taky zřejmé, že všechny odkazy jsou funkční, na rozdíl od předchozího případu firemního webu. Dle velikosti uzlů lze rovněž vyčíst, že obsahy odkazujících článků jsou téměř stejně velké.

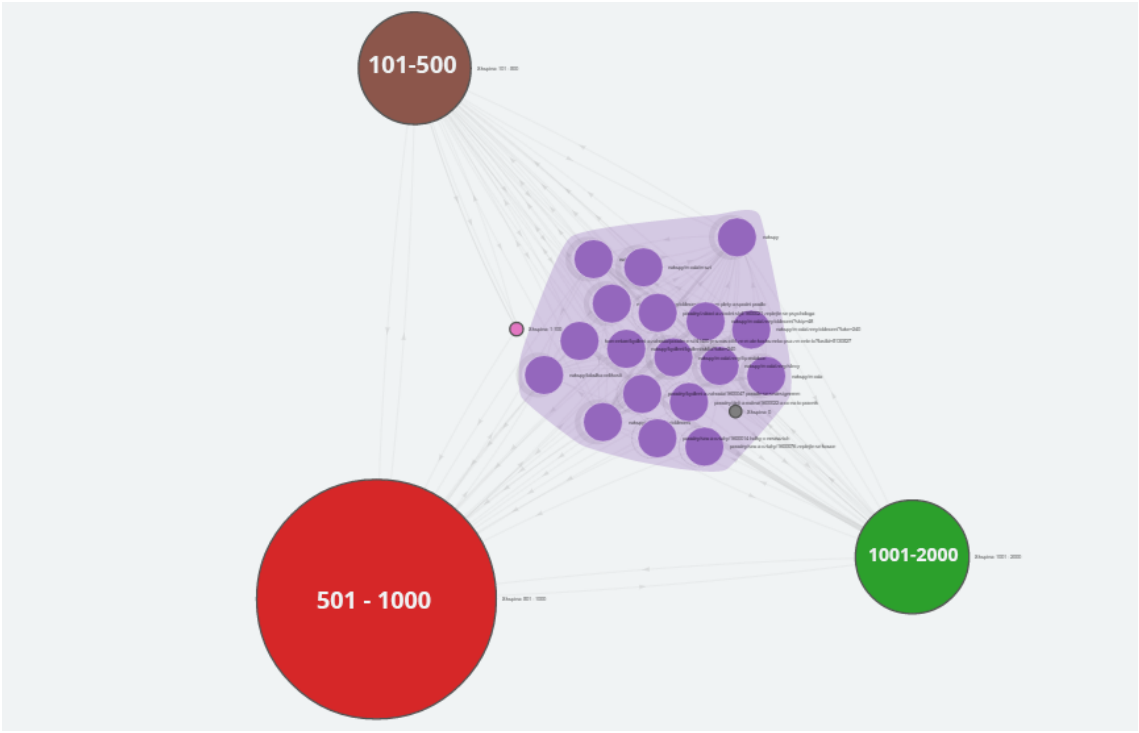
[19]Odkaz na článek http://www.prozeny.cz/magazin/bydleni-a-zahrada/poradme-si/41400-je-u-vas-citit-ze-mate-kocku-nebo-psa-zmente-to

Následně byl vytvořen nový pohled nad stejným vstupním bodem (článkem v sekci magazín - bydlení), ovšem s hloubkou zanoření dva. Bylo nalezeno celkem 400 uzlů jež propojovalo 12 448 hran.

Obrázek 32: Vizualizace webu online magazínu - hloubka zanoření 2

Velké množství hran mezi uzly vedlo k poměrně hustému uspořádání uzlů v síti. Výpočet optimálního layoutu grafu tak trval déle a jak je vidět na obr. 32, algoritmus již nebyl schopen poskytnout tak kvalitní výsledky jako v předchozích případech. K řešení tohoto problému lze využít implementované filtry, kterými lze rozštěpit rozsáhlou síť na menší. Ve vizualizaci šlo i přes velkou hustotu uzlů identifikovat jeden nefunkční odkaz, viz oranžová barva. Pro hlubší analýzu bylo využito seskupování uzlů podle různých kritérií. Seskupení uzlů, např. podle počtu slov vedlo k odhalení, že prohledané stránky mají nejčastěji obsah v rozsahu 501 až 1000 slov. Porovnáním obrázků 32 a 33 lze vidět, jak velký vliv mělo seskupení uzlů na samotnou vizualizaci.

Bylo zjištěno, že v případě významného nárůstu počtu uzlů a hran, který může nastat hlubší analýzou webu, se vizualizace stává více nepřehlednou. Kromě toho se zvyšujícím se počtem zobrazených elementů se v rámci nástroje zvýšily i nároky na výkon. Při jeho nedostatku se odezva vizualizace velmi prohlubuje. Aby se tyto požadavky snížily, bylo nutné optimalizovat vizualizační nástroj. Uzly a hrany byly dosud vykreslovány pomocí jazyka SVG. To znamená, že každý grafický objekt musel být obsažen v rámci DOM a držen v paměti. Snížením počtu těchto objektů by se bezpochyby odezva nástroje zvýšila. Vykreslování SVG objektů bylo tedy nahrazeno pomocí HTML 5 Canvas. Jelikož bylo nutné zachovat míru interaktivity u uzlů, Canvas se použil jen pro vykres-



Obrázek 33: Seskupení uzlů podle počtu slov

lení hran, jejichž počet byl v rámci vizualizace nejvyšší. Touto optimalizací se vizualizace stala plynulejší.

6.2.2 Zhodnocení

Na případové studii se podařilo ilustrovat, že před vizualizací větších webů je nutné se zamyslet nad úrovní zanoření crawlera a vstupní URL adresou. Bylo zjištěno, že při hlubším zanoření se vizualizace stává velmi nepřehlednou a to hlavně díky hustému propojení uzlů, které je důsledkem výskytu stejných odkazů na různých stránkách webu. Dále se podařilo vizualizační nástroj optimalizovat změnou metody vykreslování hran z původní technologie SVG na Canvas. Při analýze konkrétního článku v rámci online magazínu *prozeny.cz* se podařilo díky vizualizaci odpovědět na otázky kladené na začátku případové studie. Při hlubší analýze s dvojnásobnou hloubkou zanoření byl identifikován jeden nefunkční odkaz. S vyšší hustotou uzlů se potvrdil význam implementovaných datových filtrů a techniky shlukování.

7 Zhodnocení práce

Vizualizace v dnešní době hrají stále větší roli v mnoha oblastech. Prostředí internetu aktuálně nabízí mnoho podpůrných technologií, které tvorbu vizualizací umožňují. Na základě těchto technologiích vzniklo množství knihoven, aplikací a služeb, které může programátor nebo i laik využít k vizualizaci vlastního datového souboru. Jelikož prostředí webu je spíše interaktivní, vnikají i interaktivní vizualizace. Ty přináší uživatelům možnost data zkoumat z různé perspektivy a hloubky. Vizualizace již nejsou nudné obrázky, ale sofistikované aplikace, které se někdy mohou podobat hrám. Pomocí některých vizualizačních knihoven lze dokonce hry vytvořit. Příkladem může být knihovna *D3.JS*, která byla v této práci využita při vizualizaci webu. Kromě nově vzniklých vizualizačních technik vnikly i nové postupy při zpracování dat. Data generována z různých zdrojů exponenciálně přibývají a jejich zpracování se stává klíčovým tématem nejen pro mnoho společností. Zpracováním webových dat se již několik let zabývá např. společnost Google. Jejich nástroj Google Analytics je schopen téměř v reálném čase mapovat činnost uživatele na webu. Přitom zpracovává mnoho měření, na jejichž základě může majitel webové prezentace své stránky optimalizovat. Kromě toho webové stránky navštěvuje skoro každý den různé množství automatizovaných robotů - crawlerů s jiným účelem. Může se jednat o robota indexujícího obsah webové prezentace pro účely vyhledávačů, nebo robota, jehož hlavní činností je zmapovat potenciálně bezpečností chyby. Po odhalení těchto chyb lze např. webovou stránku napadnout různými typy útoků. Na internetu se tedy kromě běžně surfujících uživatelů mohou vyskytovat i jiné entity.

Vizualizace a zpracování dat má smysl, a to i dat internetových. Webové stránky jsou jasným příkladem toho, že kolem nás existují data, jejichž zpracováním lze získat zajímavé informace. Jako konkrétní příklad může posloužit tato práce a vytvořená aplikace, kde se smysl vizualizace v prostředí webu a zpracování webových dat projevil především na otázkách kolem:

- Automatizovaného získávání dat - návrhem a implementací automatizovaného robota pro sběr informací o struktuře webu.
- Možnostech vizualizace v prostředí webového prohlížeče - popisem dostupných technologií jako je HTML 5 Canvas, SVG a WebGl.
- Vytvoření vlastní vizualizační techniky - vytvořením a implementací konceptu vizualizace struktury webové prezentace pomocí knihovny D3.JS.
- Interaktivní stránky vizualizace - návrhem a implementací různých interaktivních prvků.
- Využití dalších podpůrných moderních technologií jako je framework AngularJS, ASP.NET MVC či knihovny SignalR pro vzájemnou komunikaci mezi klientem a serverem.

7.1 Problémy při vývoji

Při řešení ukázkové aplikace bylo ve fázi konceptu důležité vhodně navrhnout celkovou podobu vizualizačního nástroje tak, aby uživatel získaná data jednoduše pochopil. To se podařilo, ovšem s nárůstem hran a počtu uzlů se vizualizace stávala dle očekávání nepřehlednou. Bylo to způsobeno hlavně díky hustému propojení odkazů v rámci stránky, které se často opakovaly a vyskytovaly na různých místech. Tento problém se podařilo vyřešit implementací různých filtrů a zároveň shlukováním uzlů do skupin, např. podle počtu slov, obdrženého HTTP statusu, či dle meta tagu *robots*. Kromě větší nepřehlednosti, nárůst počtu vykreslovaných prvků měl vliv i na výkon aplikace, a to především díky využití jazyka SVG pro vykreslování. Vykreslování hran bylo tedy realizováno pomocí HTML 5 Canvas. Jelikož bylo nutné zachovat jistou míru interakce, zůstalo vykreslování uzlů zachováno. Tímto kombinovaným způsobem vykreslování se celkově zvýšil výkon aplikace, a to hlavně díky eliminaci velkého počtu SVG elementů v rámci HTML DOM.

7.2 Další možnosti rozšíření a využití

Vytvořenou aplikaci lze, díky jejímu návrhu, poměrně jednoduše rozšiřovat. Funkčnost crawlera spolu s datovým modelem lze upravit, např. rozšířením podpory detekce pozice odkazů i na jiné sémantické elementy nebo napojením na další externí analytické služby. Kromě zmíněného, díky implementaci vizualizačního nástroje postaveného na knihovně D3.JS, je možné klientskou část aplikace dále rozšiřovat o další interaktivní prvky a grafy s minimem úsilí. Aplikace může najít uplatnění např. v různých internetových řešeních umožnujících správu obsahu webových stránek. Alternativně by mohla v budoucnu sloužit jako veřejně dostupná služba.

8 Závěr

V této práci jsem se zaměřil na možnost vizualizace webových dat s cílem využít dostupné moderní technologie a postupy v této oblasti. Zmapoval jsem možné techniky sběru a reprezentace dat různých dimenzí. Dále jsem popsal vizualizační a interakční techniky, které jsem následně využil k navržení konceptu vizualizace struktury webové prezentace. Na základě navrženého konceptu byla vytvořena ukázková aplikace.

Na ukázkové aplikaci byl ilustrován celý proces vizualizace dat. Pomocí automatizovaného robota (crawlera) implementovaného na platformě .NET frameworku byla zpracována a uložena data o struktuře dané webové prezentace. Tato data byla současně doplněna informacemi o obsahu prohledávaných stránek. Vytvořený datový model umožnil vizualizovat strukturu webu pomocí síťového grafu. Získaná data bylo možné svázat s daty z externího analytického nástroje Google Anylytics. Využití ukázkové aplikace v praxi bylo demonstrováno na dvou případových studiích.

Využití moderních webových technologií umožnilo vytvářet složité interaktivní vizualizace dostupné odkudkoli bez ohledu na platformu. Tento fakt potvrzuje, že vizualizace na webu má v dnešní době velký význam.

Bc. Martin Haščák

9 Reference

[1] DOLÁK, Ondřej. Big data: Nové způsoby zpracování a analýzy velkých objemů dat. Ekonomické a informační systémy v praxi [online]. 2011 [cit. 2015-03-30]. Dostupné z: http://www.systemonline.cz/clanky/big-data.htm

[2] 3Vs (volume, variety and velocity). ROUSE, Margaret. Computer Glossary [online]. 2013 [cit. 2014-04-27]. Dostupné z: http://whatis.techtarget.com/definition/3Vs

[3] ČERNÝ, Michal. Knihovna: Big data a jejich možnosti v kontextu knihoven [online]. Praha: Národní knihovna ČR, 2013 [cit. 2015-04-19]. ISSN 1801-3252. Dostupné z: http://knihovna.nkp.cz/knihovna131/131104.htm

[4] KEIM, Daniel A. Information visualization and visual data mining. In: IEEE Transactions on Visualization and Computer Graphics [online]. 2002, s. 1-8 [cit. 2015-04-10]. ISSN 10772626. DOI: 10.1109/2945.981847. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=981847

[5] KEIM, Daniel A. Visual Techniques for Exploring Databases. In: KOPS - The Institutional Repository of the University of Konstanz [online]. 1997 [cit. 2015-04-21]. Dostupné z: http://kops.uni-konstanz.de/handle/123456789/5675

[6] KEIM, D.A. a H.-P. KRIEGEL. Visualization techniques for mining large databases: a comparison. In: IEEE Transactions on Knowledge and Data Engineering [online]. 1996, s. 923-938 [cit. 2015-04-21]. ISSN 10414347. DOI: 10.1109/69.553159. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=553159

[7] CHANG, Kai. Parallel Coordinates. Syntagmatic [online]. 2015 [cit. 2015-04-21]. Dostupné z: https://syntagmatic.github.io/parallel-coordinates/

[8] ELMER, Martin. The Trouble with Chernoff. In: MapHugger [online]. March 03, 2013 [cit. 2015-04-10]. Dostupné z: http://maphugger.com/post/44499755749/the-trouble-with-chernoff

[9] KULKA, René. Europe: British spend most online hours, Turkey leads in page impressions per visit. In: E-Mail Marketing Tipps [online]. July 3, 2012 [cit. 2015-04-21]. Dostupné z: http://www.emailmarketingtipps.de/2012/07/03/europe-british-spend-most-online-hours-turkey-leads-in-page-impressions/

[10] KEIM, Daniel A. Journal of Computational and Graphical Statistics. Pixel-oriented Visualization Techniques for Exploring Very Large Databases [online]. 1996, č. 5 [cit. 2015-02-05]. Dostupné z: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.3.4078

[11] CORTESI, Aldo. Malware. Cortesi [online]. 2012 [cit. 2015-04-21]. Dostupné z: http://corte.si/posts/visualisation/malware/index.html

[12] What is a Treemap?. Spotfire Technology Network [online]. 2012 [cit. 2015-04-22]. Dostupné z: http://stn.spotfire.com/spotfire_client_help/tree/tree_what_is_a_treemap.htm

[13] Internet Usage in the European Union. Internet World Stats [online]. © 2001-2015 [cit. 2015-04-21]. Dostupné z: http://www.internetworldstats.com/stats9.htm

[14] BOUSH, Sam. New Media Marketer. In: Website Information Structures [online]. 2011 [cit. 2015-02-10]. Dostupné z: http://www.newmediamarketer.com/website-information-structures/

[15] W3C Wiki. HTML structural elements [online]. 2014, last modified 29 December 2014 [cit. 2015-02-23]. Dostupné z: http://www.w3.org/wiki/HTML_structural_elements

[16] KESSIN, Zachary. Programming HTML5 applications. 1. ed. Sebastopol, Calif: O'Reilly Media, 2011. ISBN 978-144-9399-085.

[17] Getting started with WebGL. Mozilla Developer Network [online]. © 2005-2015, Jan 9, 2015 [cit. 2015-04-22]. Dostupné z: http://www.developer.mozilla.org/en-US/docs/Web/WebGL/Getting_started_with_WebGL

[18] WebGL Fundamentals. TAVARES, Gregg. HTML5 Rocks [online]. 2012 [cit. 2015-04-22]. Dostupné z: http://www.html5rocks.com/en/tutorials/webgl/webgl_fundamentals/

[19] How to visualize very large networks. CAMBRIDGE INTELLIGENCE. KeyLines - Network Visualization Software [online]. 2014 [cit. 2015-02-25]. Dostupné z: http://keylines.com/network-visualization/visualize-large-networks

A Obsah přiloženého CD

Na přiloženém CD se nachází kompletní zdrojový kód aplikace. Kromě toho se zde nachází ukázkové video demonstrující funkčnost vytvořeného vizualizačního nástroje. V následující tabulce je uvedena struktura přiloženého CD.

Adresář	**Popis**
/SRC/	Zdrojový kód aplikace
/VIDEO/	Ukázkové video demonstrující provoz aplikace
/DOC/	Návod pro spuštění aplikace
/TEXT/	Kompletní text práce